TEST INFRASTRUCTURE DESIGN FOR ADC CORES IN SYSTEM-ON-CHIP

Peter Mrak

Doctoral Dissertation
Jožef Stefan International Postgraduate School
Ljubljana, Slovenia, June 2010

Evaluation Board:

Assist. Prof. Dr. Anton Biasizzo, Chairman, Jožef Stefan Institute

Prof. Dr. Gorazd Kandus, Member, Jožef Stefan Institute

Prof. Dr. Andrej Žemva, Member, Faculty of Electrical Engineering, University of Ljubljana

MEDNARODNA PODIPLOMSKA ŠOLA JOŽEFA STEFANA
JOŽEF STEFAN INTERNATIONAL POSTGRADUATE SCHOOL

Peter Mrak

TEST INFRASTRUCTURE DESIGN FOR ADC CORES IN SYSTEM-ON-CHIP

Doctoral Dissertation

SNOVANJE TESTNE INFRASTUKTURE ZA JEDRA ANALOGNO-DIGITALNIH PRETVORNIKOV V SISTEMU V ČIPU

Doktorska disertacija

Supervisor: Prof. Dr. Franc Novak

Ljubljana, Slovenia, June 2010

Index

Abstract

With the advent of System-on-Chip (SoC), intense research efforts are being carried out in order to find efficient practical solutions for testing deeply embedded, mixed-signal cores. Due to restricted access, a built-in self-test (BIST) of mixed-signal cores is in many cases a preferred alternative to a conventional test using external automatic test equipment (ATE). BIST solutions to the problem of an analog-to-digital converter (ADC) test have been developed and practiced over the past years; their application in a SoC test infrastructure, however, remains an open issue.

In the ADC test, a histogram-based test and an oscillation-based test are the two approaches most suitable for BIST implementations.

The histogram-based test involves the application of an analog signal to the ADC input and a record of the number of times each code appears on the ADC outputs. These recorded samples are then used with theoretical samples in a complex computation to determine the ADC parameters, i.e., offset, gain, differential and integral non-linearity.

In an oscillation-based test (OBT), the circuit-under-test is converted into an oscillator and possible faults are assumed to manifest in the oscillation frequency. The OBT method has been practiced mostly in the area of different classes of filters. It has been only rarely applied in practice for testing ADCs.

Implementations of the histogram-based test and the oscillation-based test of ADCs in a test wrapper in a SoC are related to the problems of the minimization of the BIST overhead, test time and the achieved measurement accuracy. So far, most of these problems have been addressed only theoretically. The lack of reported experimental evidence encouraged our work on the design of a histogram-based test technique in a IEEE Std 1500 wrapper together with a thorough evaluation of the implemented test infrastructure on experimental case studies. We implemented three extreme versions of the histogram-based BIST with regards to the test-time/hardware-overhead trade-off and scalability. The IEEE Std 1500 test wrapper with the above BIST structures was implemented in a Spartan3 XC3S200 FPGA. The MAX165 (an 8-bit microprocessor compatible ADC) was initially chosen as the unit-under-test. The measured parameters of the employed ADC were in conformance with the specifications provided by the manufacturer, which demonstrates the feasibility of the histogram-based test technique with the implemented test structures.

So far, most of the OBT-related work has been directed either towards the problem of modification of a given circuit-under-test into an oscillator, or towards the analysis of detected faults (exploring ways to increase fault coverage or even trying to perform a fault diagnosis). Little attention has been paid to the measurement accuracy of the developed OBT solutions. Our feasibility studies of OBT-based BIST performed on a simulation environment using Matlab Simulink revealed an inherent measurement uncertainty of the approach. The measurement of the oscillation period is precise, but due to the arbitrary input signal phase shift and with respect to the ADC conversion two oscillation periods are possible. This in turn manifests itself as a measurement uncertainty. We further elaborated this issue and derived a theoretical background for computing the measurement uncertainty of the approach.

Povzetek

V sodobni elektroniki se vedno intenzivneje uveljavljajo integrirana vezja, ki vsebujejo množico različnih (digitalnih, analognih in mešanih) komponent v istem ohišju. Z razvojem takšnih sistemov v čipu se je pojavila potreba po razvoju primernih praktičnih rešitev za testiranje vgrajenih analogno-digitalnih jeder. Zaradi kompleksne strukture sistema v čipu so močno omejene možnosti dostopanja in komuniciranja do vgrajenih jeder. Zaradi tega se velikokrat namesto klasične metode testiranja, z avtomatskimi testnimi napravami, uporabi vgrajene samodejne teste. Že vrsto let obstajajo rešitve za vgrajene samodejne teste analogno-digitalnih (AD) pretvornikov. Za vgrajene AD pretvornike v sodobnih več jedrnih sistemih pa ni zaslediti praktičnih rešitev, sploh pa ne takih, ki bi bile v skladu z zahtevami standardov.

Za preizkušanje vgrajenih AD pretvornikov se najpogosteje uporabljata histogramska in oscilacijska metoda.

Histogramska metoda vključuje uporabo analognih signalov za vzbujanje in obdelavo števila pojavitev posameznih kod na izhodu AD pretvornika. Število pojavitev posameznih kod AD pretvornika se primerja s teoretično izračunanimi referenčnimi vrednosti glede na obliko vhodnega signala. Na podlagi primerjave se potem določi statične karakteristike AD pretvornika in sicer zamik, ojačenje, diferencialna in integralna nelinearnost.

Oscilacijska metoda temelji na konfiguriranju testiranega vezja v oscilatorsko vezje. Morebitne napake se odražajo na spremembi frekvence osciliranja. Oscilacijska metoda se pogosto uporablja za testiranje filtrov različnih razredov. Uporaba oscilacijske metode za testiranje AD pretvornikov se redko uporablja v praksi.

Pri načrtovanju vgrajenih samodejnih testov v sistemu v čipu, na podlagi histogramske in oscilacijske metode za AD pretvornike, želimo doseči čim manjšo porabo površine, čim krajši čas in čim večjo natančnost. Obstoječe predlagane rešitve so le teoretične in večinoma preverjene samo v simulacijskih okoljih. Pomanjkanje eksperimentalnih dokazov je bil dovolj velik motiv za razvoj novih testnih postopkov na podlagi histograma.

Testni postopki, ki smo jih razvili so v skladu s standardom IEEE Std 1500. Vgradili smo jih v Spartan3 XC3S200FPGA. Za testni subjekt smo uporabili 8 bitni, mikroprocesorski AD pretvornik, MAX165. Z zgrajeno testno strukturo smo izvedli številne laboratorijske meritve na podlagi katerih smo preizkusili in potrdili ustreznost ter uporabnost metod. Dobljene meritve so v skladu s podatki podanimi s strani izvajalca, kar dokazuje izvedljivost in uporabnost vgrajenih testnih struktur.

Večina obstoječih oscilacijskih testov se ukvarja s tematiko kako konfigurirati ciljno testno vezje v oscilatorsko vezje in kako analizirati ter diagnosticirati napake. Precej manj pozornosti je bilo usmerjeno v določanje merilne natančnosti oscilacijske metode. Izvedli smo študijo o izvedljivosti oscilacijskih vgrajenih testov z zgrajenimi simulacijskimi orodji v programskem okolju Matlab Simulink. Obsežna študija je pokazala na inherentno napako oscilacijske metode testiranja AD pretvornikov. Oscilacijska metoda testiranja je natančna, toda zaradi poljubnega faznega zamika vhodnega signala in delovanja AD pretvornika rezultirata dve različni oscilacijski frekvenci. Zaradi nedoločljivosti izmerjene oscilacijske frekvence metoda vsebuje napako, ki se je ne sme prezreti pri načrtovanju testnih postopkov za AD pretvornike. Izvedli smo temeljito študijo omenjenega pojava in določili teoretične ozadje za izračun napake oscilacijskega testa.

Abbreviations

ABM	=	analog boundary module
ADC	=	analog-to-digital converter
ATE	=	automated (or automatic) test equipment
AWG	=	arbitrary waveform generator
BIST	=	built-in self-test
CFI	=	core functional input
CFO	=	core functional output
CMOS	=	complementary metal-oxide semiconductor
CPU	=	central processing unit
CTI	=	core test input
CTO	=	core test output
CUT	=	circuit under test
DAC	=	digital-to-analog converter
DC	=	direct current
DFT	=	design for testability
DLE	=	differential linearity error
DM	=	detector module
DNL	=	differential non-linearity
DPS	=	digital power supply
DSP	=	digital signal processor
DUT	=	device under test
EM	=	exploitation module
FFT	=	fast Fourier transform
FPGA	=	field-programmable gate array
FSR	=	full-scale range
IC	=	integrated circuit
ID	=	identification data
ILE	=	integral linearity error
INL	=	integral non-linearity
I/O	=	input/output
LSB	=	least significant bit
MOS	=	metal oxide semiconductor
OBT	=	oscillation-based test
OBIST	=	oscillation built-in self-test
PCB	=	printed circuit board
PMU	=	parametric measurement units
RAM	=	random-access memory
SoC	=	system-on-chip
SNR	=	signal-to-noise ratio
SQNR	=	signal-to-quantization-noise ratio
TAP	=	test access port
TBIC	=	test bus interface circuit
TDI	=	test data input
TDO	=	test data output

THD	=	total harmonic distortion
TPG	=	test pattern generator
TUE	=	total unadjusted error
UTP	=	unit test period
UUT	=	unit under test
VM	=	volt meter
WD	=	waveform digitizer
WBR	=	wrapper boundary register
WBY	=	wrapper bypass register
WIR	=	wrapper instruction register
WSI	=	wrapper serial input
WSO	=	wrapper serial output
WSP	=	wrapper serial port

1 Introduction

Testing is used in engineering, science, manufacturing, business, etc. in order to evaluate the perfection of work, products, knowledge, etc. A test is a definitive procedure that produces a test result. The test result obtained with the precision measuring instrument or simply with personal observation is meant for a qualitative, categorical, or quantitative estimation.

The test in engineering is commonly used to improve products. To keep improving the products the two steps should be continually repeated: test – redesign. Two points are relevant. First, the test should be designed in such a way that it finds the errors. Second, the redesign must correct them otherwise this procedure can be repeated indefinitely. Here we also have the question of perfection: when is the product good enough to stop testing? This question is usually linked with the cost and the time available. The go/no go test is common in production testing. The result of go/no-go testing tells us whether the device under test is good and can be used for the function meant or it is bad and not appropriate for use.

The test in electronics is an important part of device manufacturing. By constantly increasing the scale of integrated circuits the probability of faults arises. Testing of pure digital circuits is usually defect-oriented, where the test design is primarily focused on a digital waveform. Logic ones and zeros, high and low voltages, rise and fall times are the major attributes of interest in testing digital circuits. On the other hand, testing an analog circuit considers parameters like: amplitude, bandwidth, slew rate, overshoot, settling time, phase noise, etc. The portion of analog circuits compared to digital ones in electronic devices is relatively small, however there exist a variety of different test approaches for each type of analog circuits.

Testing digital or analog integrated circuits is a complex task. Testing of an integrated circuit (IC) consisting of both digital and analog elements is consequently an even more demanding task. These kinds of circuits are called mixed-signal circuits. The mixed-signal circuits are not just a combination of digital logic and the analog circuitry [1], they actually operate with both digital and analog signals. The simplest mixed-signal circuits are analog-to-digital converters (ADC), digital-to-analog converters (DAC), programmable gain amplifiers, phased locked loops, etc. Distinguishing between the digital, analog and mixed signals circuits is irrelevant in the context of a mixed-signal test, because most of today's devices include both digital and analog circuits. Some authors, like [2], add a third type of electronic circuit, i.e., memory, also requiring its own special type of test.

ADCs are used in a wide variety of applications, including consumer electronics, communications systems, process control and real-time control systems. The analog portion of them usually interfaces to the environment; this is why ADCs are one of the most frequently used mixed-signal circuits. The testing of an ADC is typically specification oriented, where the ADC parameters are evaluated. Parameters that can be measured can be broadly divided into dynamic parameters and static parameters [3]. The static parameters (i.e., offset, gain, differential nonlinearity (DNL) and integral nonlinearity (INL)) are related to the transfer function of the ADC. The dynamic parameters (i.e., total harmonic distortion (THD), signal-to-noise ratio (SNR), effective

number of bits) describe the deformation induced on the converted signal.

In order to reduce the production cost the ADCs are, along with other circuits, integrated in system-on-chip (SoC) designs. When designing the test structures for ICs in SoC the test standards should be considered to ensure interoperability. The IEEE Std 1149.1 [4] is a collection of design rules applied principally at the integrated circuit level to alleviate the growing cost of designing, producing and testing digital systems. The IEEE Std 1149.1 [4] and its extension IEEE Std 1149.4 [5] are meant for providing similar facilities for mixed signal circuits. The mechanism for the test of digital core designs within a SoC is defined with IEEE Std 1500 [6]. While the mentioned standards guides you when designing a test implementation and how to access the UUT, the problem of a communication bottleneck between the UUT in the SoC with external test equipment, remains. This is due to the high-scale integration of deeply embedded cores. A built-in self-test (BIST), which substantially reduces the data exchange between the external automated test equipment (ATE) and the UUT has been identified as a possible solution. The BIST is additional circuitry, specially designed into the IC in order to facilitate the self-testing feature [7].

The testing of the static performance parameters of an ADC in a conventional way requires an external ATE and a device interface board. The ATE supplies a suitable input stimulus via a device interface board to the ADC; the responses are captured and returned back to the ATE for processing [8]. A BIST of ADCs deeply embedded in a SoC removes the need for high-quality test equipment. The BIST performs the major part of the test activity within the UUT and only the final results are communicated out.

In the proposed work we try to develop BIST structures for embedded mixed-signal cores in a SoC. The BIST structures are based on well known test techniques (such as the histogram-based test and the oscillation-based test) but require modifications due to the limitations imposed by embedding mixed-signal cores in a SoC.

The aim of the proposed thesis is to develop BIST structures for ADC cores in a SoC. In this regard, the work will be focused on the histogram-based test and oscillation-based test techniques. Their implementation within the SoC test wrapper will be explored. The objective is to obtain efficient solutions in terms of the test-overhead/test-time trade-off. Attention will be paid to the issues of measurement accuracy. In addition, conformance to the design-for-test standards, such as IEEE Std 1149.4 [5] and IEEE Std 1500 [6] will be pursued.

There are different test methods for testing ADC static performance parameters: the edge and center code test, the step and binary search method, the feedback-loop test (also referred as the servo loop method), the histogram test method and the oscillation-based test. In reference to [8] there are mainly two ADC test methodologies adapted and used for BIST: the oscillation-based test and the histogram test method.

The histogram-based test involves the application of an analog signal to the ADC input and the record of the number of times each code appears at the ADC outputs. The count of occurrences is then compared with theoretical values in order to compute the ADC static parameters.

In the feedback-loop test an input voltage of the ADC under test oscillates around the desired code transition. This is achieved by employing the analogue integrator at the ADC input, which continuously integrates with a positive or negative slope, depending on a comparison result between the ADC output code and the desired code [9]. By measuring the average voltage at the ADC input we can get the voltage value of the particular code transition level. A very similar approach [10] is the oscillation-based test (OBT) where the parameters of the ADC are calculated from the oscillation frequency.

The doctoral dissertation is divided up as follows. First, the essentials of electronic

testing are presented, where the tradeoff among test efficiency, test time and test cost is pursued. The conventional types of IC test and test environments are introduced. Then we address the mixed-signal circuits and mixed-signal test. The mixed-signal basic test approaches and limitations are shown. In chapter four, ADC and ADC testing is described. The ADC static characteristics and methods for testing them are considered in detail. Furthermore, the histogram-based ADC test for embedded ADC cores is revised, with a special emphasis on the tradeoff between hardware overhead and test time. The modifications and improvements of the histogram BIST are described in chapter five, which were required to get acceptable solutions for a practical implementation. The test structure implementation in conformance with IEEE Std 1500 [6] is made and at the end of the chapter the measurement results collected from real experiments are presented and commented on. In chapter six, OBT of ADC is presented in a similar way, with the test structure and the environment set up in Matlab. Simulations are carried out in order to determine the ADC test accuracy and test efficiency of the OBT technique. In chapter seven, the generation of optimal diagnostic test sequences for a SoC with complex embedded cores is addressed.

The following materials are included in the Appendix. The BIST solutions of the histogram-based test and the experimental results of different implementation approaches are described in [11]. The measurement conditions of the OBT-based BIST are analyzed and a theoretical framework for the computation of the measurement uncertainty is presented in the paper [12] published in the ETRI Journal. Some selected case studies that were performed in order to assess the applicability of a diagnostic tool developed at the Jožef Stefan Institute in SoC fault localization are described in the paper [13], published in the International Review on Computers and Software.

2 Essentials of electronic testing

2.1 Test economics and product quality

The electronics industry is involved in almost every type of product you can think of. More and more products include electronic parts – just think of your car, telephone, television, radio, children's toys, etc. Furthermore, not only has the amount of electronics in these products increased, so has the complexity of implemented electronics. With the integration of electronic circuits we gain more functionality in the final product. At the same time, this reduces the reliability of high-technology electronic products. An electronic circuit lifetime of three years is typical. Electronic designers and manufactures must develop more complex, more reliable and in time to market products. One way to improve efficiency, productivity and quality is to test.

From the economic point of view, testing presents additional cost. If we could ensure that every manufactured electronic board is designed correctly, meets all its specification, and is assembled and soldered properly, we could eliminate testing. In a real world with no testing we would rapidly go out of business because our field-service costs would be astronomical (they are probably high enough even with all the testing we do) and because of a terrible reputation for poor quality and reliability. Testing therefore must be considered as a cost-avoidance strategy.

The economic optimization of the production cycle does not necessarily mean performing any operation in the cheapest way [14]. For example, if you purchase the cheapest components and consequently most unreliable, which are then installed in your product, the fault occurrence in the forthcoming production stages may be catastrophic. It is generally accepted that the earlier in a production stage a fault is found the lower are the costs. The 10-to-1 rule [14] describes how the cost to locate and repair a fault is increased about ten times at each subsequent testing stage in production.

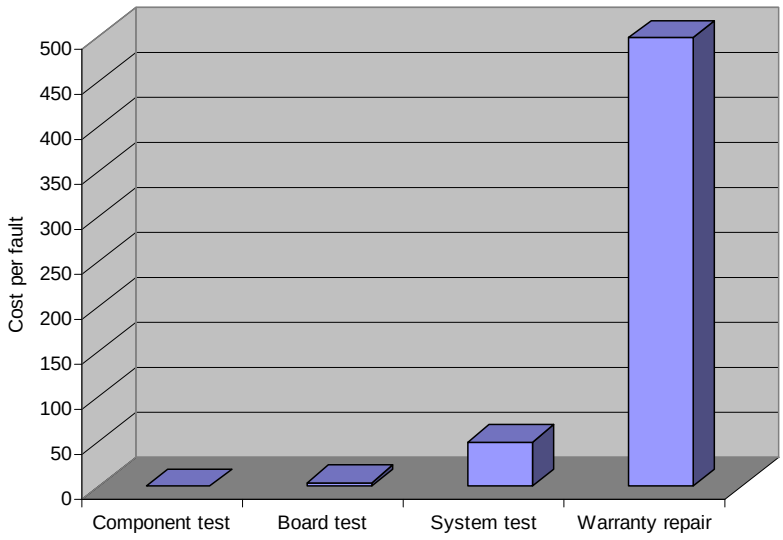


Figure 1: The 10-to-1 rule of the increasing cost of a fault at the various stages of test.

On the other hand, performing all the testing that could possibly be done is as costly as not doing enough testing. The tradeoff between test and cost must be optimized. Deciding which test, test equipment, test method is most suitable is a difficult task, since the complexity of electronic circuits is increasing. The famous Moore's law [15] predicts that the number of transistors placed on IC doubles approximately every two years. While even its author admits that it cannot continue forever because we are approaching the fundamental barrier of the size of atoms with respect to the size of transistor [16], experts expect the law to hold at least for two or three decades. However, the original Moore formulation is not just about number of transistors on IC; it predicts the trend of semiconductor manufacturing, what we can do with computers in modern life and also the cost variation of chips. Regardless of the future ambiguity of transistor count it is sure that the testing difficulty and test costs will prolong with the development.

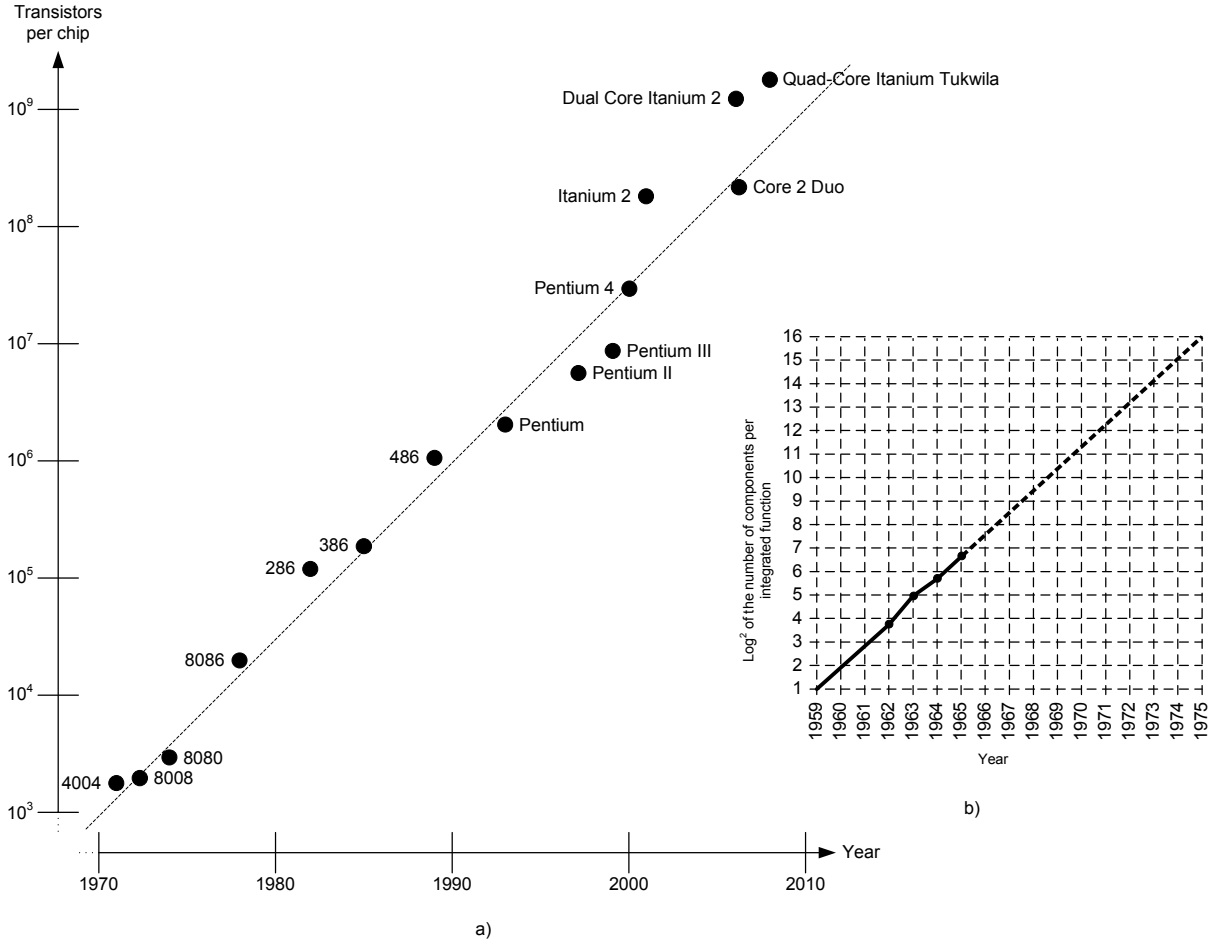


Figure 2: Moore's law a) describing the long-term trend in the history of computing power, b) original formulation from Gordon Moore, co-founder of the Intel Corporation.

It is not only the increasing density of transistors that impacts on testing problems. Rent's rule (1) describing the relationship between the modules within a chip, the number of logic blocks in modules and the signal connections [18] indicates the problems emerging from the growing number of internal interconnections. Rent's rule provides basic assumptions of interconnection density and its impact on the controllability and observability of an IC module under test. An IC module contains one or more blocks and their connections. A block is a primitive logical circuit (e.g. logical gates, a storage element, register, etc.). The signal connections are specified in terms of the number of pins.

$$P = AK^r \tag{1}$$

where *P* is the average number of pins per module, *K* the average number of blocks per module, *A* average number of pins per block and *r* the Rent exponent. Rent exponent is typically between 0.5 and 0.8. Rent's rule presents the basis for considering defect impact on power, area, delay and other relevant parameters of IC structures.

Unfortunately, even if all the possible testing is performed at the end of the production line, the reliability of the device still cannot be assured. Reliable devices must meet their specification over its life time. Measuring the reliability of a device is performed by testing during its operating time, as depicted in Figure 3.

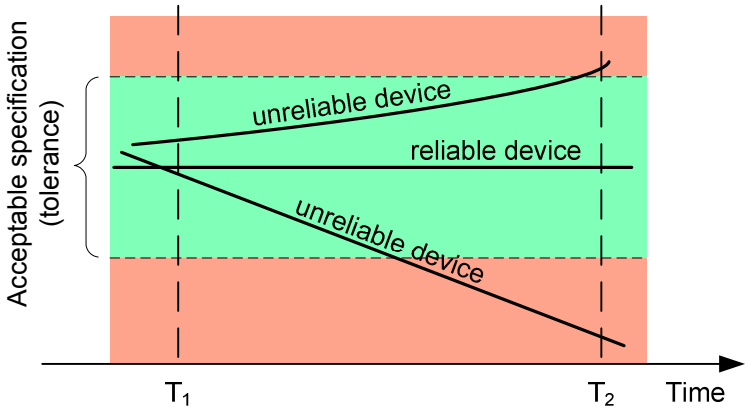


Figure 3: a device's reliability.

The time in which an unreliable device fails to operate within its tolerance may vary from several seconds to several months or even several years. A manufacturer cannot afford to test every device in a production line during longer time spans to verify its reliability. Furthermore, some faults only arise when a device is powered up several times or under certain operating conditions. The failure mechanisms can be accelerated by stressing the device. Stressing is a process where a device is put under some kinds of mechanical, thermal, electrical or chemical shocks. The procedure, which consists of testing, stressing and re-testing, is commonly called a screening program.

The failures during screening in general fall into two major areas:

- fault caused by handling of the device during test,
- early-life failures.

The second of these two areas is also often referred as infant mortality. Early-life failures are usually caused by some defects that did not show up during initial device testing, but get worse during the device operation. The failures rate at the operating stage are lowered and again increased at the end of the lifetime. This is described with the well-know bathtub curve [14]. The bathtub curve is de facto in the operating time of products.

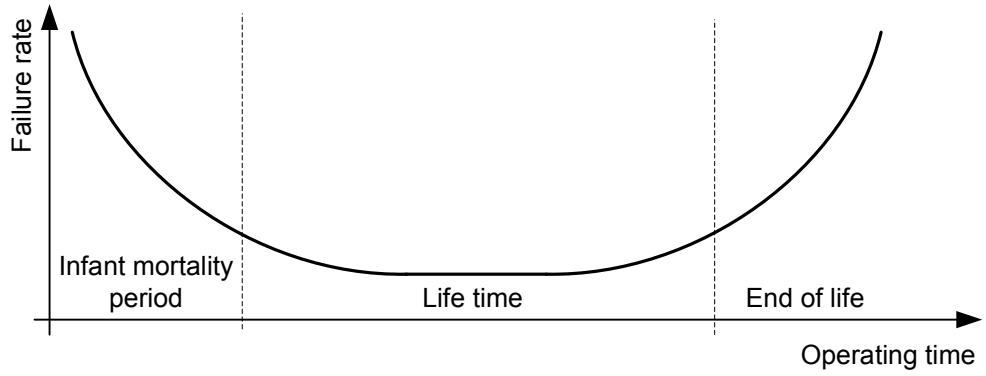


Figure 4: The bathtub curve.

The optimization of resources used for testing is crucial to minimize the overall costs. The two extreme cases can be considered:

1. no testing (no cost of test),
2. 100% test (no cost of failures).

As mentioned before, neither of the two scenarios can be employed. The emphasis is on test efficiency. The optimization should consider minimizing the test costs and maximizing the productivity, ensuring the same or greater quality and reliability. To achieve these three objectives the following steps need to be accomplished:

- selection of the optimum testing strategy or strategies,
- selection of the most cost-effective equipment,
- effective management and control of the testing department.

Proper test optimization should also consider the test time. A long test time will also prolong the time-to-market. Time-to-market is the time needed to get a product from the idea to the market place. In a fast expanding industry such as electronics it is a key point to minimize the time-to-market. This time-to-market must be considered through a thorough planning and resource management. Consequently, in the process of developing products the test costs and the test time cannot be overlooked.

2.2 Types of test

Selection of the optimum testing strategy depends a great deal on the device under test and its function. In general, digital, analog and mixed signal tests are subjected into two types of test:

- functional test,
- structural test.

As the name describes, a functional test is trying to verify the function of a device under test (DUT) or circuit under test (CUT). On the other hand, a structural test is technology dependent. While the functional test characterizes the operation of the DUT as a complete part, the structural test emphasizes the function of the individual parts of the DUT.

A more fundamental classification can be made with regard to the type of circuits tested for:

- digital (logic) test,
- memory test,
- analog and mixed-signal test.

In analog and mixed-signal tests the functional test approach is prevalent. Structural testing in general requires suitable fault models and the use of fault simulations. In contrast to the digital circuits there are no common fault models and consequently widely accepted fault simulators. Fault modeling in analog and mixed-signal are usually based on resistive open/short fault or minimum/maximum values at specific pins of the circuit components. Fault modeling rarely considers the representative range of values (i.e. voltage levels, frequency span,...). Due to an inability to test for specific analog parameters, the structural test in the analog and mixed-signal domain is limited. The boundary between functional and structural is not strictly defined. The same test method or technique can be applied as a functional or structural test at different levels of the circuit (component test, board test, system test,...).

2.3 Test environments

Choosing the appropriate test environment depends upon the company's experience in test engineering. In general there are two options:

- building your own external test environment,
- buying automated test equipment (ATE).

The first option requires a lot of expertise in test engineering and measuring techniques. Usually, the companies use this option when there are special test requirement which are not met by the available test equipment on the market. Building an ATE on its own would take considerable time and cost for companies that are not especially involved in the test and measurement industry.

Purchasing an ATE requires a thorough consideration of the testing needs, quality, reliability, and cost. Modern ATE prices are calculated in millions of dollars and that is why a bad choice would be a very expensive burden. According to the Semiconductor Industry Association roadmap [19] these costs are growing and will continue to do so.

The cost of the ATE is also one of the main reasons why the new design for testability (DFT) techniques were fostered. At increasing densities of ICs we are facing a problem of accessing the test points in a printed circuit board (PCB) or a SoC. DFT design places some additional test control circuitry in order to enable a more effective test during device production.

The conventional ATE is a system used to generate a test stimulus, acquire a test response, and analyze the test result. The ATE applies an input stimulus to the DUT and then analyzes the responses. The test result can simply identify DUT as good or bad, or gives a parametric value of the DUT.

ATE systems are external devices and use the so-called test heads, through which the ATE is connected to the DUT. ATEs are used to test a wide range of electronic devices and systems, from simple components (resistors, capacitors, and inductors) to ICs, printed circuit boards (PCBs), and a complex SoC.

A complex ATE that is capable of testing a SoC is controlled by a workstation or personal computer (PC) and usually has several central processing unit (CPU) or even one CPU per pin in order to provide fast data processing and a data-reduction capability.

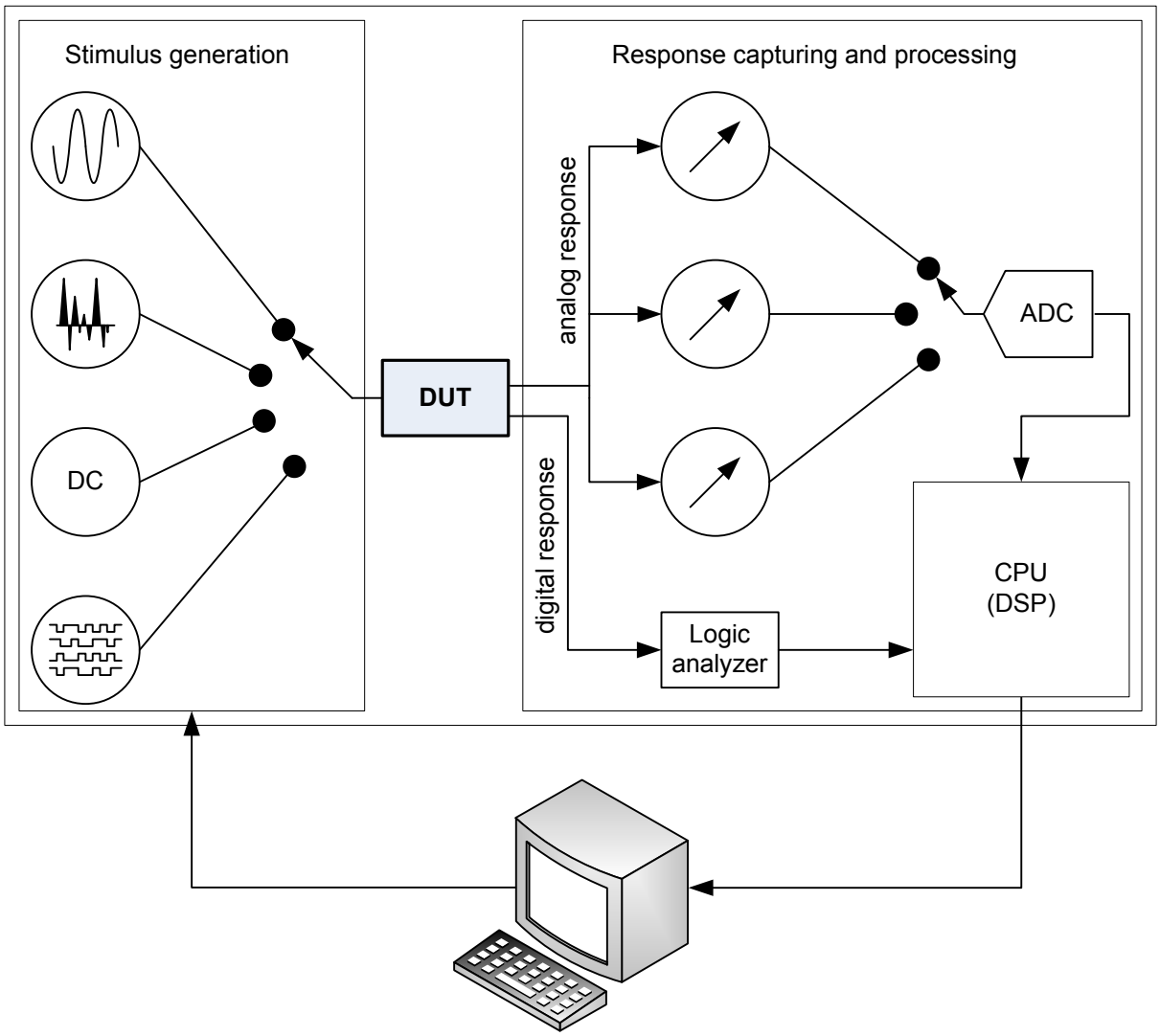


Figure 5: Concept of automated test equipment (ATE).

A complex ATE consists of several instruments; including an arbitrary waveform generator (AWG), parametric measurement units (PMUs), a test patterns generator (TPG), a digital power supply (DPS), etc. All of these instruments are high precision, therefore ATEs are extremely expensive. With the development of the digital signal processor (DSP) modern ATEs are usually DSP-based testers.

The DSP systems measure, filter and compress continuous analog signals. The analysis is performed in digital form once a signal and its components have been converted to numbers. A signal in digital form is easier to analyze, rearrange and processing than the analog form.

Signal analysis is usually preformed by the DSP. In digital signal processing the signal characteristics in the time and frequency domains are evaluated. The DSP is a special-purpose type of CPU which has implemented algorithms for analyzing signal data. The algorithms that a DSP carries out are ultra-fast instruction sequences, such as shift and add, which are used in math-intensive signal-analyzing applications. The DSP in a test system can be used in order to perform measurements. The characteristic values which are obtained with DSP can be compared to the expected values to evaluate the processed data.

The DSP-based testing gives several advantages, such as:

- reduced test time; it can create and measure signals with multiple frequencies at the same time,
- signal components separation; isolation of noise and distortion components from one another and from the test signal enables more accurate and repeatable

measurements,

- coherent sampling; ensures good repeatability and error elimination.

The DSP testing is advantageous when multiple measurements must be made. On the other hand, there are a large number of computations required to perform, which leads to a complex and costly DSP. The accuracy of the DSP-based test mainly depends on the accuracy and precision of the stimulus generators and the ADC which digitalizes the test signal. An ADC or DAC inserts inevitable quantization noise. To avoid the impact of quantization noise, the number of converter bits must increase, which consequently leads to an increase of hardware and the test time.

The DSP-based test involves three steps:

1. Stimulus generation; the DSP controls the stimulus generator, which generates a digital or analog signal. Analog signals are produced by an arbitrary waveform generator (AWG). Digital signals are stored in memory and sent to the DUT.
2. Response capturing; if the DUT responds with the analog signal, the waveform digitizer (WD) converts and stores the response. If the DUT returns a stream of digital data, that response is stored in a capture memory.
3. Response analyzing; Stimulus generation and response capturing last for a predefined period of time. When data capturing has finished the DSP processes the received data.

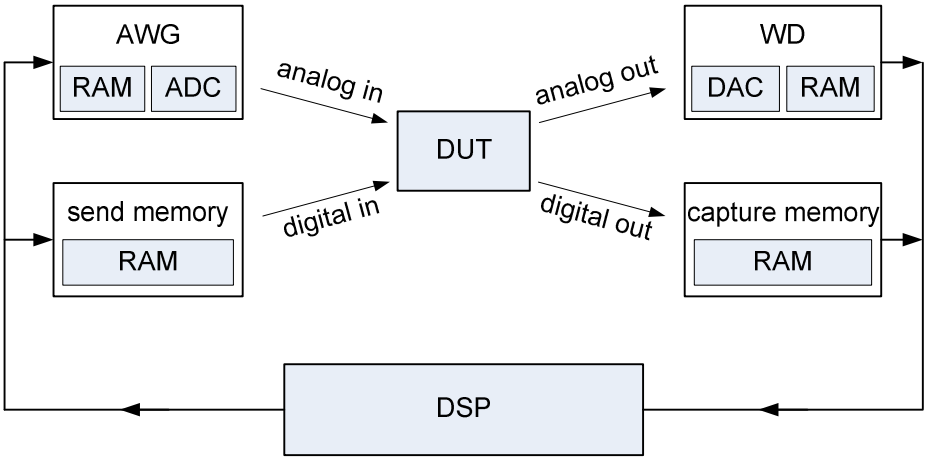


Figure 6: The DSP-based test system.

2.4 Design for testability

While the conventional ATE test environment is very effective, its costs have become a problem. With the introduction of surface-mounted devices, in-circuit test techniques utilizing a bed-of-nails to make contact with each individual DUT on a printed circuit board is almost impossible. There is also a problem with the communication bottleneck because these devices usually operate at very high frequencies. Due to the increasing logic-to-pin ratio there is a problem of accessibility with deeply embedded cores in a SoC. These problems fostered the development of design-for-testability (DFT) methods which provide a cost-effective, high-quality, at-speed test. At the beginning the purpose of DFT was to develop and apply manufacturing tests for designed hardware. These structural test solutions were mostly designed for digital circuits and the result was usually “go/no go” to ease the communication between the CUT and external tester.

Over the years the industry developed a variety of detailed and standardized DFT approaches. Initially the DFT techniques primarily focus on good design practices learned from experience also called ad-hoc DFT. Ad-hoc DFT design tries to eliminate situations which cause poor controllability and observability, for example:

- asynchronous logic feedbacks,
- non-initializable flip-flops,
- redundant gates,
- gated clocks,
- difficult-to-control signals, etc.

Ad-hoc DFT requires experts and tools that are not always available. Furthermore, newer circuits have become too large for manual inspection and once the testability problems are found redesign is necessary. Due to these problems, the use of ad-hoc design is usually discouraged for large circuits. With the increase in the complexity and size of circuits an alternative form of DFT was developed, known as structured DFT. The most popular structured DFT methods are scan design, boundary-scan and built-in self-test (BIST).

2.4.1 Scan design

The digital logic systems can be divided into combinational and sequential circuits. There are several methods of designing combinational circuits that are easy to test. The generation of a test sequence for sequential circuits is confronted with two additional difficulties: first, the initial state of the circuit is unknown and second, the propagation of the fault to the primary output depends on the state of the circuit. These problems arise due to the memory elements (flip-flops) in the sequential (digital) circuits.

The scan design's main objective is to obtain the control and observability of flip-flops. This is achieved by organizing all flip-flops into one or more shift registers. The flip-flops can now be initialized via serial data shift-in line. Next, the circuit is put in its normal operation for a few clock cycles and the resulting internal states of flip-flops (i.e., test results) are shifted out via the serial data shift-out line. While there are various implementations of the concept, the main idea is to partition the logic into combinational and sequential parts and to add the scan path to the flip-flops. With the scan design the test of complex digital circuits becomes feasible for mass production.

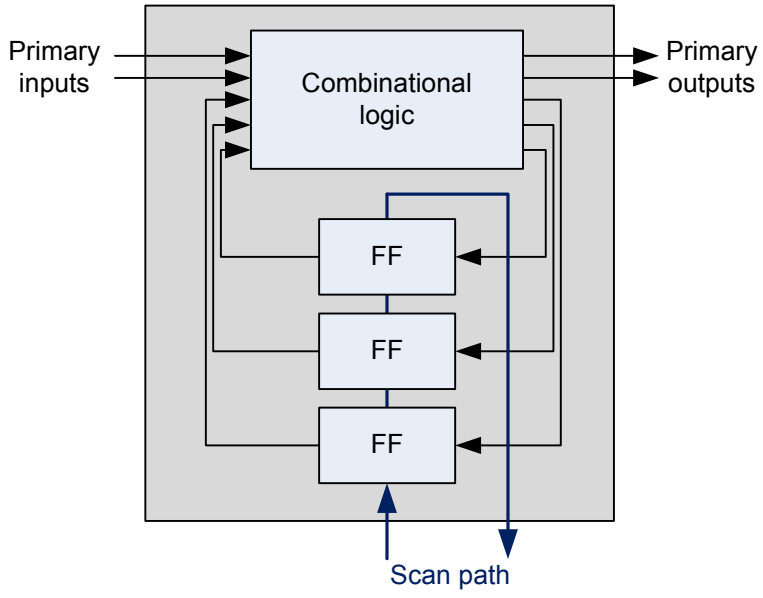


Figure 7: A scan design schematic.

2.4.2 Test standards and boundary scan

The aim of standards is to insure consensus of interest from different designers and developers. Standards include definitions and terminology, methods of measurement and test, structure guidelines, temperature limits, applications guidelines, recommended practices, safety consideration, etc. They include good engineering practice. Consensus requires that all views and objectives are considered, but are not limited to. Standards result from technical considerations by different competent organizations. Standards are some kind of substantial agreement reached by directly and materially affected interest categories. A substantial agreement presents much more than just a majority of interest categories, but not necessarily unanimity.

The IEEE standards provide a common ground for communication in some specific areas of electronic technology. The IEEE standards also ensure criteria for measuring the acceptable performance of circuits, equipment or materials related to the specific field of electronics. The IEEE organization is a leading authority on areas such as aerospace systems, computers and telecommunication, biomedical engineering, electric power, and consumer electronics.

The test designs presented in the following sections combine the standard design rules in order to provide testable devices. In the case of placing different cores from different providers on a single chip, individual cores can be tested without compromising intellectual property. The testing standards make it possible to access the internal pins of IC cores without the need for a physical connection, as presented in Figure 8.

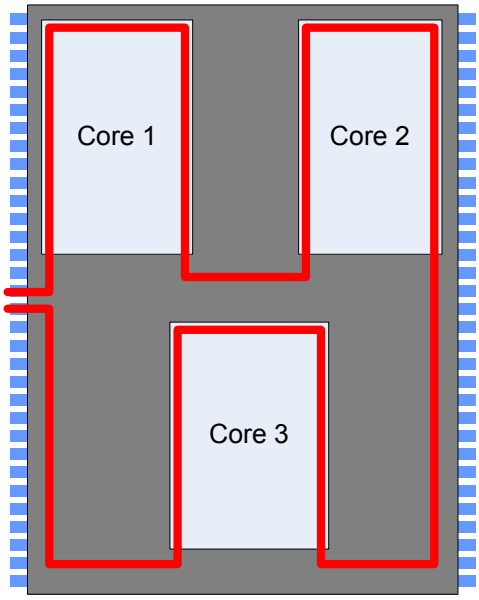


Figure 8: Design for a testability chip with different IC cores.

The IEEE Std 1149.1 [4] standard test access port and boundary-scan architecture informally known as boundary-scan consist of a collection of design rules applied principally for an IC. Initially it was used for testing printed circuit boards using boundary scan. A fundamental benefit of the boundary scan is in complex designs were testing problems are the most difficult. It transforms the extremely difficult testing problems of IC into well-structured problems. Test software can be easily programmed when circuits are designed in conformance with the boundary scan. In this way different test data can be applied to the CUT. Over the years the boundary scan also becomes a tool for accessing sub-blocks of IC and a mechanism for debugging embedded systems that may not have any other debug-capable communications channel.

The IEEE Std 1149.1 informally known as boundary scan is a special scan path with a register cell added at every I/O pin on the device circuitry. The added register cells are

organized as a serial scan path providing access to the I/O pins of individual IC. In this way, the technique improves the controllability and observability of the unit under test. The basic architecture (Figure 9) consists of an instruction register, a bypass register, device ID register, a boundary register (boundary register cells) and a test interface, referred to as the test access port (TAP).

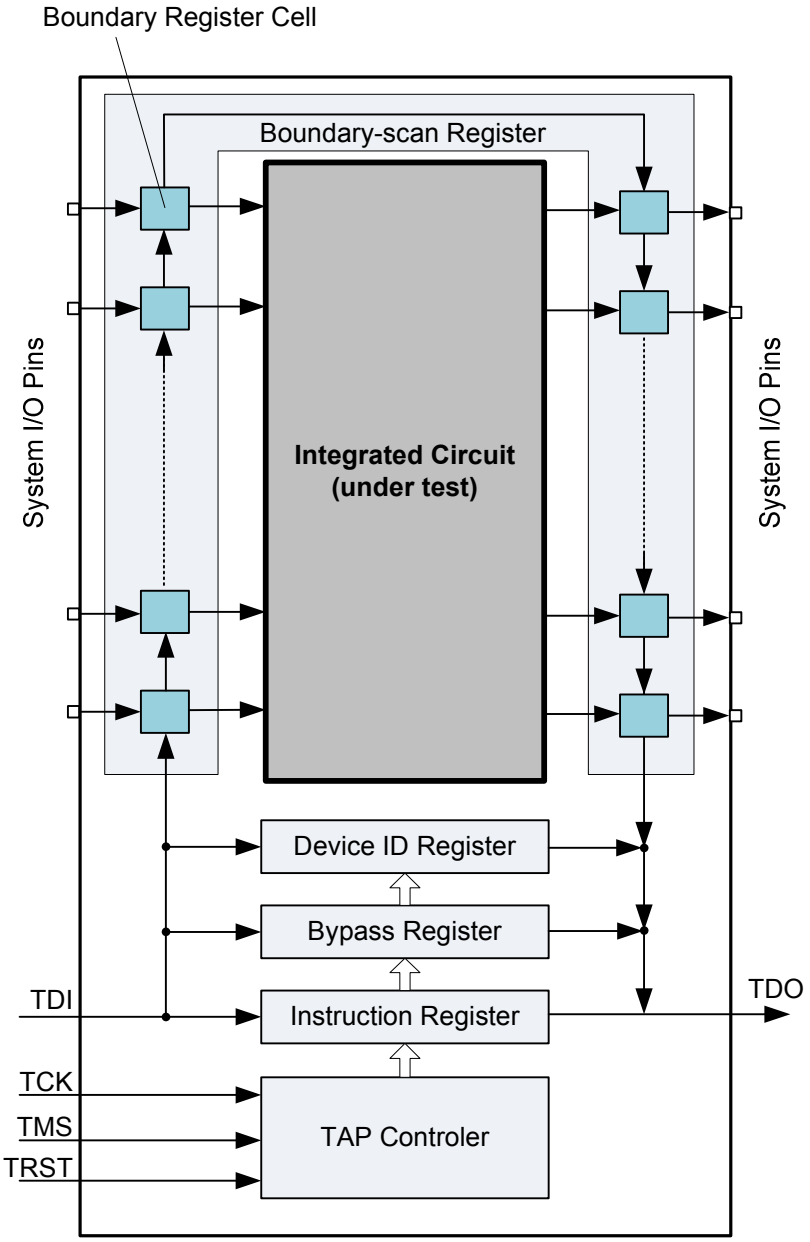


Figure 9: The general architecture of the IEEE Std 1149.1 compliant IC.

As depicted in Figure 9, five new pins are dedicated to the circuitry. The test data input (TDI) and the test data output (TDO) are used for serial shifting data into and out of the IC. The other three pins from the TAP controller are used with a simple protocol to communicate with the boundary scan on-chip circuitry. These pins cannot be shared with any other function.

The main advantage of the boundary scan testing standards is that they can be used by PCB designers, IC designers and systems designers when implementing different technologies, without the need to fully understand the testing problems of the particular technology. The usage of the boundary scan test port greatly facilities a built-in self-test (BIST), especially when implementing BIST at different levels of packaging (chips, boards, systems).

2.4.3 Built-in self-test

Advances in semiconductor technology have enabled the design of complex SoCs consisting of deeply embedded digital and mixed-signal cores, which makes their testing a difficult task. The communication bottleneck and the increasing gap between the operating frequencies of an external ATE and the SoC under test led to the transfer of the SoC testing task from the external ATE to an internal built-in self-test (BIST). BIST solutions reduce the test access requirements and the costs of test equipment.

A BIST is a mechanism that permits a system to test itself. The definition of BIST is (as defined in the Sematech Official Dictionary, on-line version at http://www.sematech.org/publications/dictionary/):

“Any of the methods of testing an integrated circuit (IC) that uses special circuits designed into the IC. This circuitry then performs test functions on the IC and signals whether the parts of the IC covered by the BIST circuits are working properly.”

The BIST technique is used in wide variety of products. In ICs, BIST is used to make faster and less expensive tests. From the beginning it was mainly used to enable the circuit to test itself and determine whether it is good or bad at the end of production line. With the recent development of embedded systems BIST becomes a viable test technique at all levels of design (it can be used for wafer and device-level testing, for all manufacturing testing and for system-level testing) providing also the measurement of different parameters.

In order to achieve a self-test capability the circuits require additional hardware and functionality, which is incorporated into the design. The additional hardware and functionality must ensure an appropriate test-pattern generation and a mechanism for processing the output responses to detect faults.

The BIST introduces many advantages such as a reduced need for external testing, at speed testing, more economical testing, reduced time-to-market, etc. The main disadvantages of BIST are hardware overhead, additional design and time efforts, and performance penalties (for instance, increased power consumption).

3 Mixed-signal test

The signals in real life, science and engineering (time, distance, temperature, lightness, etc.) are analog. Analog-to-digital and digital-to-analog conversion allow computers to interact with these signals. The parts of devices which interact with both kinds of signals are called mixed-signal circuits.

A mixed-signal circuit operates across digital and analog domains. It can process either analog or digital information in either analog or digital form. For the simplest mixed-signal circuits, we can consider a complementary metal oxide semiconductor (CMOS) switch. In this circuit the digital signal controls the analog resistance of a CMOS transistor. Because of the trivial interconnection between the analog and digital domains, this circuit is not always considered to be mixed-signal circuit at all. The most common mixed-signal circuits are analog-to-digital converters (ADC) and digital-to-analog converters (DAC). As the names indicate these circuits convert the continuous analog signal into a digital representation (number) and vice versa, as shown in Figure 10.

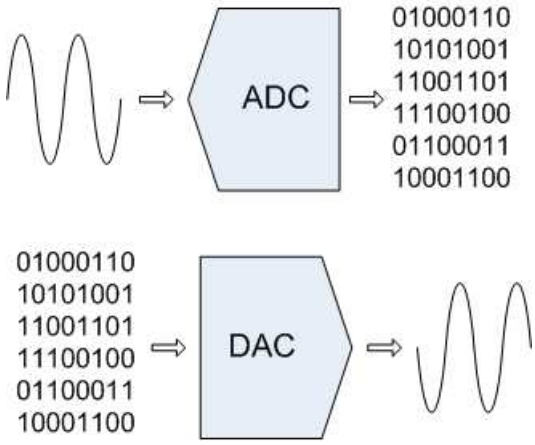


Figure 10: Data converters.

While an analog signal has numerous parameters and information, after the conversion the signal contains only sampled and quantized parts of analog information. The sampling frequency, the number of bits and the type of filtering define the amount of information after the conversion. The testing of mixed-signal circuits poses a number of unique problems from the digital and especially from the analog part. Testing the digital part is usually defect-oriented, while analog tests are specification or parametric oriented, expressed in terms of a range of signal values resulting from environmental variations (temperature, humidity and ageing) and noise. Assuming that the digital portion of mixed-signal circuits can be tested using the existing digital test techniques, the mixed-signal test problem focuses mainly on analog test problems [20].

The basic differences between analog and digital testing are summarized in Table 1, from which we can see why the analog (consequently mixed-signal) testing is a more demanding job than digital testing.

Table 1: Basic differences between analog and digital testing

	Digital	Analog
Test signal:	Discrete (binary)	Continuous
Response analysis:	Direct interpretation	Mathematical post-processing
Measurement equipment:	Low precision	High precision
Fault model:	Mainly catastrophic	Catastrophic and parametric
Fault-free response:	Binary vector	Tolerance interval in multidimensional space

3.1 Sampling, reconstruction and quantization

The processes which convert the signal from an analog (i.e. continuous) signal to the discrete and back are sampling and reconstruction. Both sampling and reconstruction are extensively used in mixed-signal testing. The tester reconstructs the signal to stimulate the DUT and then measures the response. The mixed-signal DUT samples and reconstructs (or vice versa) the signal as a part of normal operation.

If the conditions of the sampling theorem [21] are met, a continuous waveform can be sampled and then reconstructed without the loss of signal information, as shown in Figure 11. The continuous signal is described as a continuous function of time $v(t)$, which is converted into a sequence of discrete samples $v[n \cdot T_s]$. The reconstruction operation performs an interpolation between sampled values to form the reconstructed continuous signal $v_R(t)$.

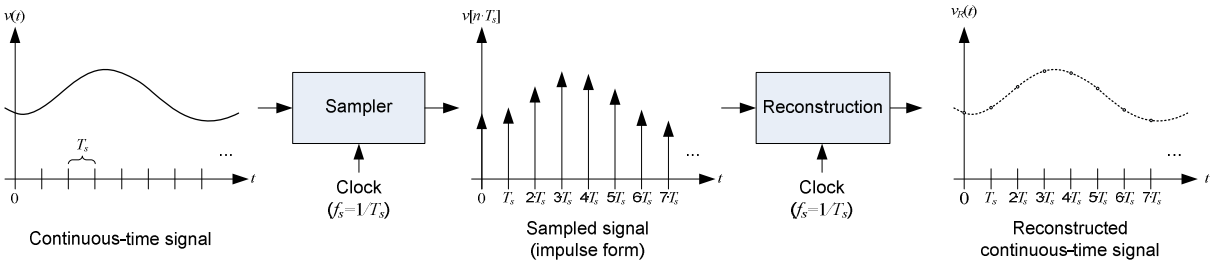


Figure 11: Sampling and reconstructing the signal waveform.

The sampling theorem was proven by Shannon [21], while the interest and knowledge of sampling in engineering applications may be traced back to Nyquist [22]. The various extensions and applications of the sampling theorem can be found in [23]. The sampling theorem for band-limited signals can be simply stated as in [1]: “A continuous-time signal with frequencies no higher than f_{max} may be completely recovered from knowledge of its samples taken at the rate of $2 \cdot f_{max}$ per second.”

A mathematical sampling can be achieved with no loss of signal quality. The sampling is considered as a quantization in time. Due to the finite precision of digital signal we must also consider amplitude quantization. The output of the quantizer (ADC) is divided into a number of equal quantization intervals. The amplitude quantization is a process where a sampled signal amplitude value is represented by a value of the quantization interval it resides in. Usually, the value representing the quantization interval is the mid-point, in some cases the quantization interval can represent the lower or the upper bound. The quantization error is the difference between the quantized (digitized) and the “true” analog value. The quantization error is defined by the number of quantization intervals and can only be avoided if the number of intervals goes to infinity, which is infeasible in practice.

3.2 Coherent sampling

Performing the mixed-signal measurements using DSP-based ATE it is typically assumed that the original signal can be restored by periodically repeating the collected samples during the unit test period (UTP). The UTP is often called the primitive period. From that follows the primitive or fundamental frequency f_p as:

$$f_p = \frac{1}{UTP}. \tag{2}$$

The UTP selection significantly affects the signal reconstruction, as depicted in Figure 12. If the UTP is a multiple of the signal period, as shown in Figure 12a, the restored signal smoothly transits over the UTP ends. The measurement schemes whose UTP is a multiple of the signal period is referred to as coherent sampling. The coherence property makes the sample locations controllable and repeatable [24]. On the other hand, if the UTP is not a multiple of the signal period, as presented in Figure 12b, signal discontinuities appear at the UTP ends.

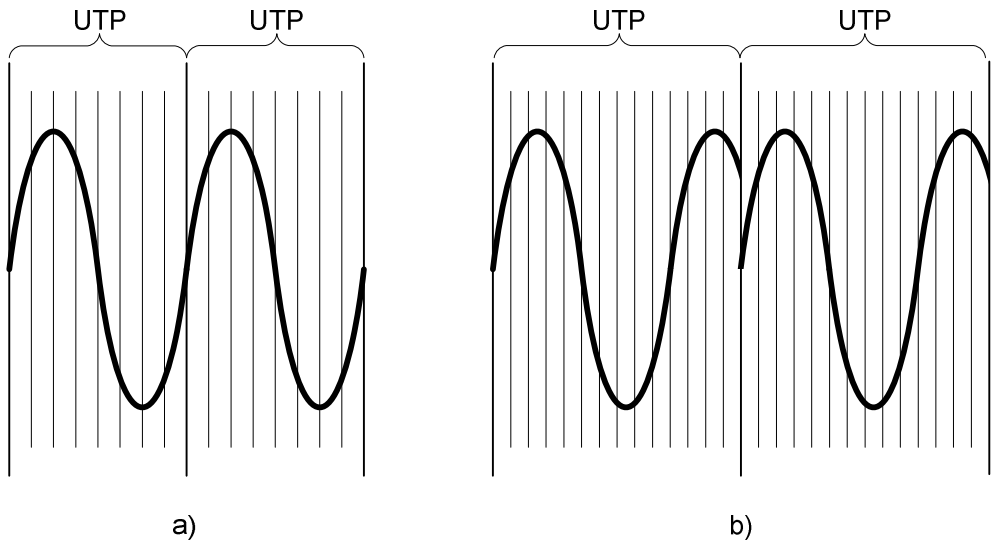


Figure 12: Signal reconstruction with a) smooth transitions, b) signal discontinuity.

In coherent sampling the UTP extends over M signal cycles and consist of N sampling intervals, where M and N are integers. The primitive frequency is related to the signal frequency f_i and the sampling frequency f_s as:

$$f_i = M \cdot f_p, \tag{3}$$

$$f_s = N \cdot f_p. \tag{4}$$

The equations (2), (3) and (4) lead to the fundamental requirement

$$\frac{f_i}{f_s} = \frac{M}{N}. \tag{5}$$

where the M and N are integers to ensure coherent sampling. If the M and N are relatively prime each sample contributes unique and independent signal information, which is the preferred measurement approach.

Coherent sampling represents the basis when setting up Fourier transforms. A waveform analysis with Fast Fourier transforms (FFT) will result in an error in the frequency spectrum results, if the coherence condition is not met.

In practice it is often not feasible to ensure coherent sampling. Different testers, test equipment, test signal generator and the DUT do not share a common clock. Consequently, test system components are not synchronized to each other and not coherent by nature [8]. In the case of non-coherent sampling we use windowing to reduce the unwanted discontinuity to a minimum. Windowing is a mathematical process that changes the signal over the UTP in such a way that it forces the signal to decay to zero at both ends of the UTP.

3.3 Accuracy, precision, error and uncertainty

When discussing measurements, results of measuring or measuring instruments there are several terms (accuracy, precision, error and uncertainty) which are often confused. In conversational English the terms accuracy and precision are usually used for identical meaning. For example [25] gives almost identical definitions of them and uses one term to explain the other. In engineering textbooks these terms are often defined with very different meanings. The following definitions are derived from different sources [1, 26, 27, 28, 29, 30, 31] and give us a technical understanding that is also meant in the following chapters:

- Accuracy – The difference between the measurement result (or average of the measurement results) and the true or correct value. Accuracy cannot be discussed unless the true value is known. Accuracy does not tell us about the quality of the measurement instrument. The instrument may be of high quality or can measure highly precisely, but may exhibit a high systematic error. The accuracy of the test instruments tells us the degree of conformance to absolute values, usually expressed as a percentage of the instruments' full scale range.
- Precision – The precision refers to the repeatability of measurements. The variation of measurement results obtained by repeating measurements under the same conditions is defined as the measurement precision. The actual value of the measurements is not important because precise test instruments give repeatable results.
- Error – The difference between the measurement result and the true value is defined as the error. Error is a single value and if known can be applied as a correction to the result. In general, errors are divided into two types: systematic error and random error. Systematic errors are those that appear consistently from measurement to measurement. Systematic errors caused by the measurement instruments can often be reduced with calibration. The random errors, as the word random indicates, are inherently unpredictable. Irrespective of using an inexpensive digital voltmeter or two million dollars worth of ATE, measurements have random errors.
- Uncertainty – The uncertainty of a measured value is an interval around that value such that any repetition of the measurement will produce a result which lies within this interval.

3.4 Measurement-error reduction

3.4.1 Filtering

Measurements are corrupted by random variations; they are affected with a noise. A filter is a device that removes certain components from a signal. Low-pass analog filters are often used to improve measurement repeatability. A low-pass filter excludes some electrical noise, since it removes signals with frequencies higher than the cutoff frequency. However, when implementing a low-pass filter in the tester hardware it takes longer to measure DC voltages. Thus there is an inherent tradeoff between the test time and better repeatability. A low-pass filter plays the same role in signal processing as averaging does in some other fields.

3.4.2 Averaging

Averaging is a form of discrete time filtering. When averaging the series of measurements the repeatability of the measurement results can be predicted. In other words, the measurement results will become more repeatable and reliable if the length of the measurement series is increased. For example, to reduce the effect of the noise by a factor of two, approximately four times more readings should be collected and averaged. From the test time point of view excessively increasing the collection of samples becomes prohibitively time consuming. There are different types of averaging used to analyze a set of data points (simple moving average, cumulative moving average, weighted moving average, exponential moving average).

3.4.3 Guard band

Setting test limits different than the datasheet specification limits introduces the risk of accepting defective units and rejecting conforming devices. Guard banding is an important technique dealing with measurement uncertainty. The measurement uncertainty affects the risk of incorrectly declaring DUT as in-tolerance or out-tolerance. In order to prevent a faulty declaration of the DUT the test limits should be narrowed for the measurement uncertainty. This is achieved by lowering the upper specification limit for the measurement uncertainty and increasing the lower specification limit, respectively. A guard band is the offset from the specification limit to acceptance test limit, as depicted in Figure 13.

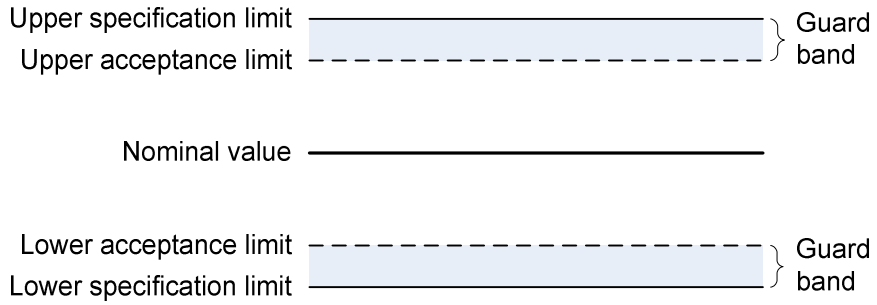


Figure 13: Guard band limitation.

In many applications the guard band and the measurement uncertainty have the same value, but fundamentally they are different issues. The measurement uncertainty as stated in the previous subsection is a statistical limit on the probable error. The guard band is a part of the calibration process designed to alter the results on a statistical basis. Setting the guard band limits influences the false-accept risk and the false-reject risk. The false-accept risk is

the probability that an out-tolerance DUT will be accepted as an in-tolerance DUT, due to the measurement error and vice-versa, the false-reject risk is the probability indicating that an out-tolerance DUT will pass the test as the in-tolerance DUT.

To set the correct criteria for the band guard, different standards and guides are available [32, 33, 34] to help test designers.

4 ADC test methods

4.1 The ideal ADC

The ADC can be considered as a true mixed-signal device, because it converts between the analog and digital domain and has both analog and digital functions. A very simple illustration of what the ADC is: a device that provides an output that digitally represents the input voltage or current level. The majority of ADCs convert an input voltage, but the definition of an ADC also includes the possibility of an input current [35]. The ADC typically uses an analog reference against which the input is compared. The result of this comparison tells us what fraction of the reference represents the input voltage. This is also the reason why the ADC is sometimes called a divider.

The ADC performs three basic functions: sampling, quantization and data coding (Figure 14). As described in section 3.1, the sampling transforms a continuous-time signal into its sampled data equivalent. Quantization changes the analog continuous-level to a discrete level. Finally the coding scheme outputs binary values on the ADC output pins.

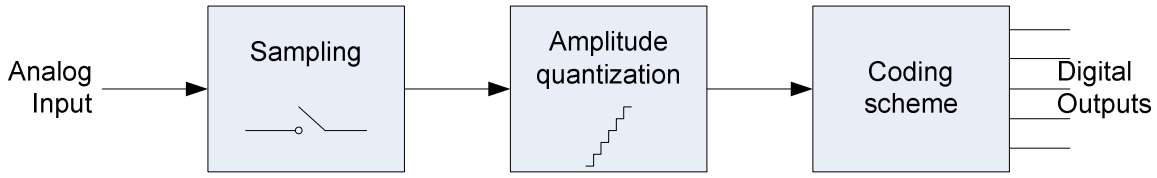


Figure 14: ADC basic functions.

The number of ADC bits denoted by n defines the number of possible quantization outcomes N. The number of ADC quantization intervals (ADC codes) is given as:

$$N = 2^n . \tag{6}$$

The resolution of the ADC is defined with the transfer curve. The ADC transfer curve has a many-to-one mapping function and it is defined as:

$$\text{output} = 2^n \cdot G \cdot \frac{V_{in}}{V_{ref}} \tag{7}$$

where G stands for gain, V_{in} for analog input voltage and V_{ref} for the reference voltage. Generally, the gain of an ADC is unity (G=1), but there are also some manufacturers that have introduced ADCs with other gain factors.

Most of the ADCs have a linear transfer function. The interval of the input values mapping to each output value have a linear relationship and the intervals are uniform for the whole input range. A non-linear transfer function introduces more information (better resolution) for certain regions of the input values.

Two definitions of the transfer curve exist with regard to the code transition points between code levels: mid-tread and mid-rise. The transfer curve is called mid-tread, where the first transition is half of the quantization interval above the minimum input voltage. When the first

transition occurs at the minimum input voltage the transfer curve is named mid-rise. The dissertation considers the static characteristic test procedure for mid-tread ADCs with a linear transfer curve, as shown in Figure 15.

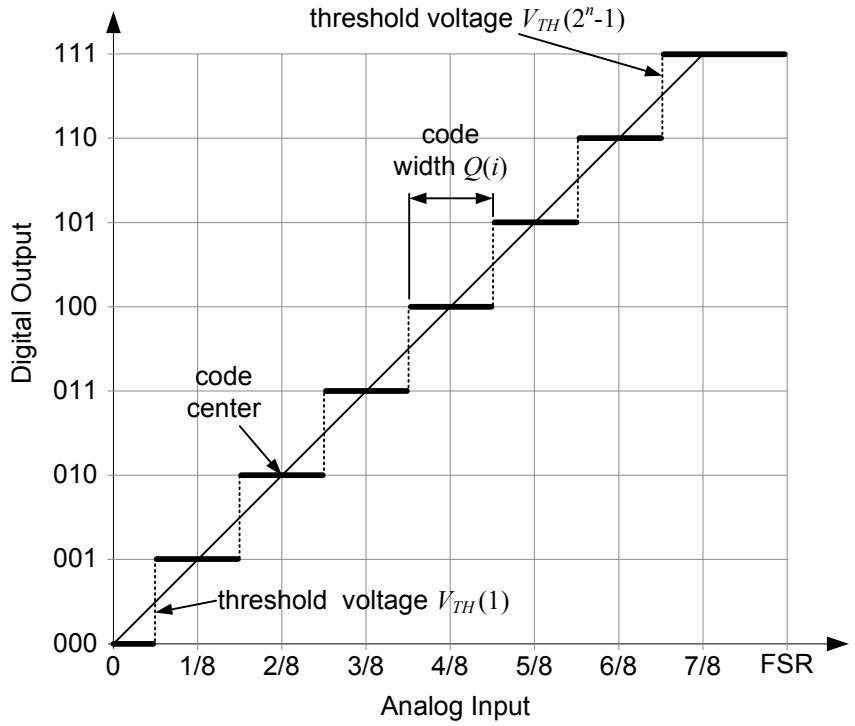


Figure 15: Ideal transfer curve of a 3-bit ADC.

An ADC generates a single output code for a range of input voltages. The full-scale range (FSR) of the ADC is determined with a reference voltage, usually $V_{ref}=FSR$.. The voltage level where the ADC output changes code (code transition) is defined as the threshold voltage $V_{TH}(i)$. The difference between the two sequential threshold voltages is called the code-width $Q(i)$. The code widths of an n-bit ideal ADC are equal and called nominal code width Q_N. Nominal code width (Q_N) is determined by dividing the full-scale range FSR by the number of ADC codes:

$$Q_N = \frac{FSR}{2^n}. \tag{8}$$

Nominal code-width Q_N is the analog voltage equivalent to one bit. The resolution of the ADC is denoted with voltage equivalent to be one bit. One bit is defined as the size of the least significant bit (LSB). The LSB has weight 1. Resolution is the fundamental limit that defines the quantization error (or quantization noise). The ADC quantization error, in the case of the uniformly distributed signal at input, varies between the -1/2LSB and +1/2LSB and it is equal to zero in the code center, as depicted in Figure 16.

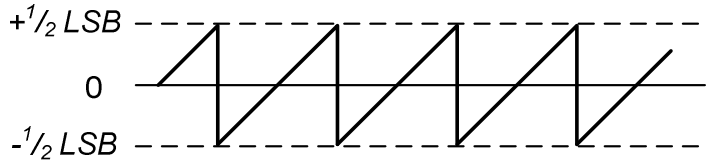


Figure 16: Quantization error.

The signal-to-quantization-noise ratio (SQNR) is a measure of the effect of quantization errors introduced by analog-to-digital conversion. The *SQNR* is measured in decibels [dB] and depends on the number of ADC bits and the shape of the input signal. In the case of a uniformly distributed input signal, such as a triangular wave, the SQNR is calculated as:

$$SQNR = 20 \cdot \log_{10}(2^n) \approx 6.02 \cdot n \quad [\text{dB}]. \tag{9}$$

In practice this means each ADC bit improves the SQNR by approximately 6.02 dB.

The properties described above present the basis of the analog-to-digital conversions, which we have to take into consideration before applying the ADC under test.

4.2 ADC architectures

The ADC type is normally defined with its conversion algorithm [36]. The most common types of ADC with regard to the conversion algorithm and structure are:

- flash ADC,
- successive approximation ADC,
- integrating ADC,
- sigma-delta ADC.

The flash ADCs performs a direct conversion (that is why they are sometimes also called direct conversion ADCs). A direct conversion means that they compare the input signal against all possible decision levels simultaneously. The flash ADC architecture consists of a large number of resistors in series (resistive divider), 2^n-1 numbers of comparators for *n*-bit ADC and a decode logic circuit (Figure 17). The decode logic circuit generates a code for each voltage range, depending on which of the comparators producing logic ones has the highest threshold voltage.

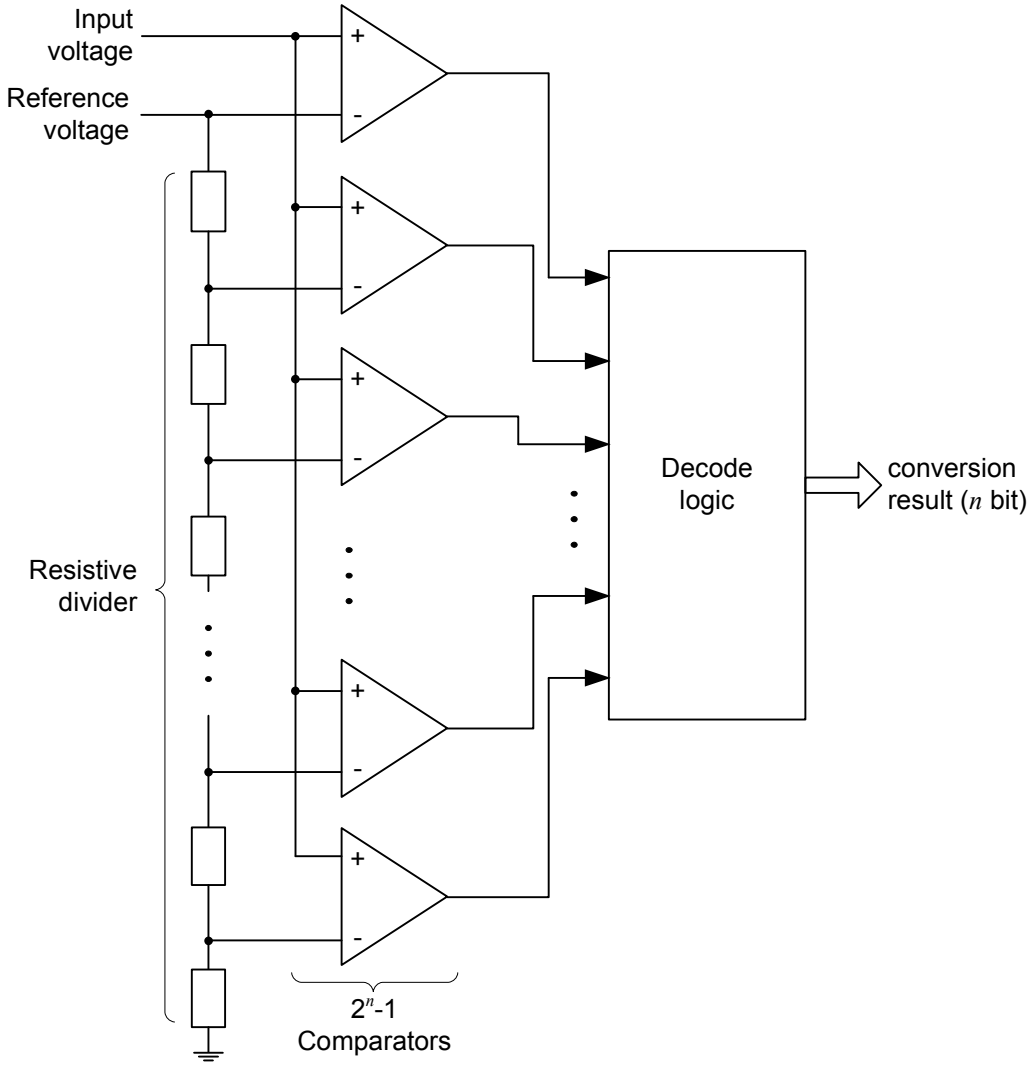


Figure 17: Flash ADC.

The direct conversion enables the flash ADC to operate at very high frequencies (capable of a gigahertz sampling rate). Because the decision levels are compared all at once, the sample and hold circuit is not required. Since the number of comparators grows exponentially with the number of ADC bits, the flash ADC becomes very expensive as the resolution increases.

The successive approximation converter is the most common type of ADC is, because it represents an acceptable compromise between speed and accuracy. This type of ADC includes a DAC, whose output is adjusted with a binary search algorithm. The comparison between the DAC output and the input voltage is performed with an analog comparator. A successive approximation register controls the DAC output, moving the output value up and down, depending on the result of the comparison. When the binary search is completed the successive approximation register's value (i.e. DAC input code) represents the output code. The successive approximation design also includes the sample and the hold circuit to maintain the input voltage at same value, while the successive approximation is in progress. The successive approximation ADC architecture is depicted in Figure 18.

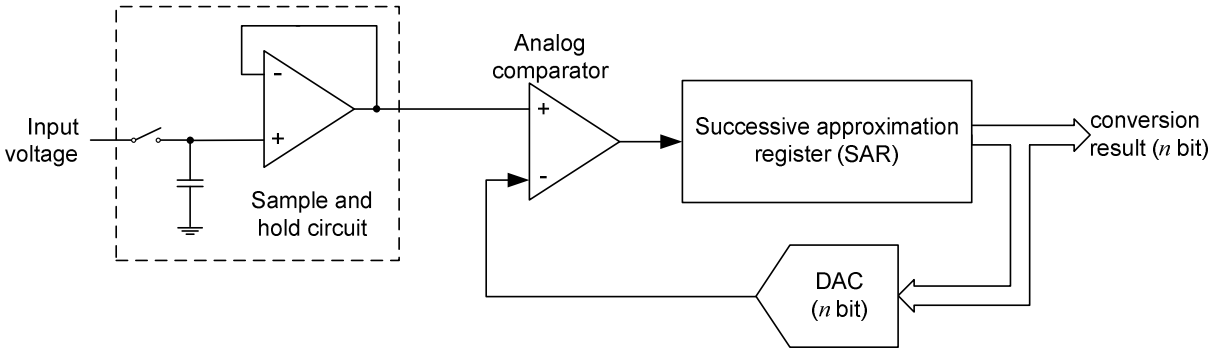


Figure 18: Successive approximation ADC.

The successive approximation ADC performance is limited by the performance of the implemented DAC, the sample and hold circuit and the analog comparator. While any type of DAC can be implemented, the problems that affect the implemented DAC also affect the ADC. In addition, the sample and hold circuit and the analog comparator may have poor linearity and hysteresis errors, which affect the characteristics of the successive approximation ADC.

The integrating ADC uses the step search algorithm, like the successive approximation ADC implies a binary search. The integration ADC has implemented an integrator, to which the input voltage is applied. The integrator then ramps up for a fixed time period (the run-up time). After the run-up time the known reference voltage in the opposite polarity is applied to the integrator, which ramps down with a fixed slope until it reaches the threshold voltage again (the run-down time). The run-down time is proportional to the integrator's peak voltage, which in turn is proportional to the input voltage. The run-down time is measured by a counter, whose output represents the ADC output. This type of converter is also called a dual-slope converter because it ramps up and down. The schematic of integrating ADC is shown in Figure 19.

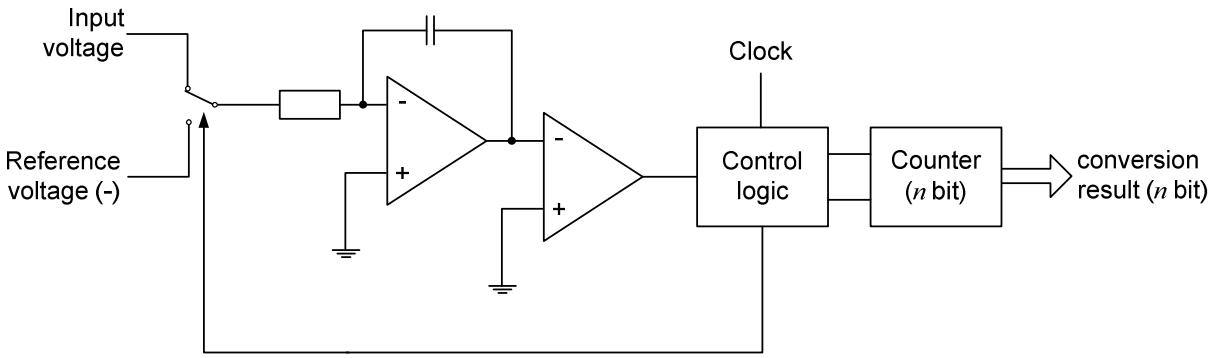


Figure 19: Integrating (dual-slope) ADC.

The integrating ADC has excellent DNL characteristics, since each code is dependent on a smooth ramping of the analog integrator. The integrator's linearity errors in the upward ramp cancel the linearity errors in the downward ramp. On the other hand, the conversion times of the integrating ADC are long due to the time needed for ramping up and down.

A sigma-delta ADC oversamples the desired signal at a frequency much higher than the Nyquist frequency (two times the input bandwidth). This is why the sigma-delta is also sometimes referred as over-sampling. Also, the words sigma and delta in the name are often used in a different order. The sigma-delta uses a modulator which consists of an integrator, a comparator and a feedback loop (1-bit DAC) to produce an oversampled, pulse density modulated data stream. This is then filtered with a digital filter to produce high-resolution ADC samples. A first-order sigma-delta ADC block diagram is presented in Figure 20.

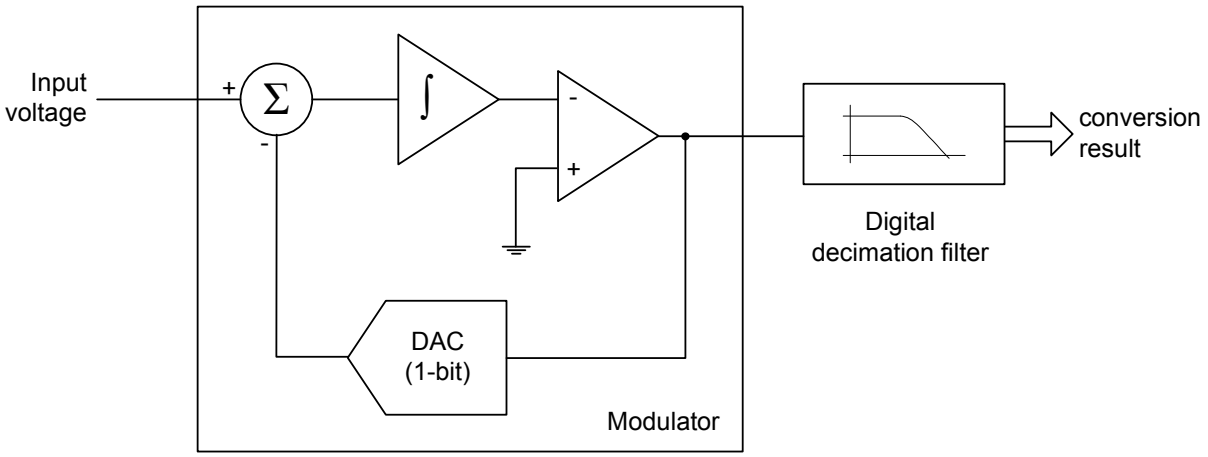


Figure 20: Block diagram of the first-order sigma-delta ADC.

To fully understand the principle of the sigma-delta conversion one must become acquainted with noise shaping, digital filters and decimation. How an oversampled sigma-delta converter works is described in detail in [37]. The simple analog side of the sigma-delta ADC and complex digital side are what makes the sigma-delta ADC inexpensive to produce. The low-cost conversion method, providing both the high dynamic range and the flexibility in converting the low bandwidth input signals, is manifested in increased production of the sigma-delta ADC.

4.3 ADC test types

Tests for ADC are, in general, functional, because of the different architectures of ADCs with large discrepancies in the structures. With regard to the ADC operation, the test can be divided into the following two categories:

- static test,
- dynamic test.

The main aim of the ADC static test is to identify the code edges (threshold voltages). For this reason the static ADC test is, in general, a code edge measurement. The threshold voltage is, in practice, the input voltage (also named the code edge voltage or the code transition voltage level) with the same probability of having two contiguous codes at the output of ADC. The static ADC characteristics (or parameters) can be obtained, after some processing, from the code edge's identification. The static (DC) characteristics of the ADC refer to the input-output relationship, defined by the transfer curve. The static ADC characteristics include offset, gain, differential and integral non-linearity, monotonicity, and missing codes.

The dynamic ADC tests consider the dynamic operation of the ADC (the effects of signal changes and frequency-related effects). The aim of dynamic performance testing is to identify the signal's components, such as the conversion time, the signal-to-noise ratio (SNR), the total harmonic distortion (THD), the effective number of bits (EONB), etc.

A general test structure for the ADC test is depicted in Figure 21. The basic setup for static and dynamic testing is the same; obviously the speed of the source generator and the digital processing capabilities must be adequate for testing the target ADC. For example the dynamic test environment employs a FFT to determine the spectrum of the ADC output response.

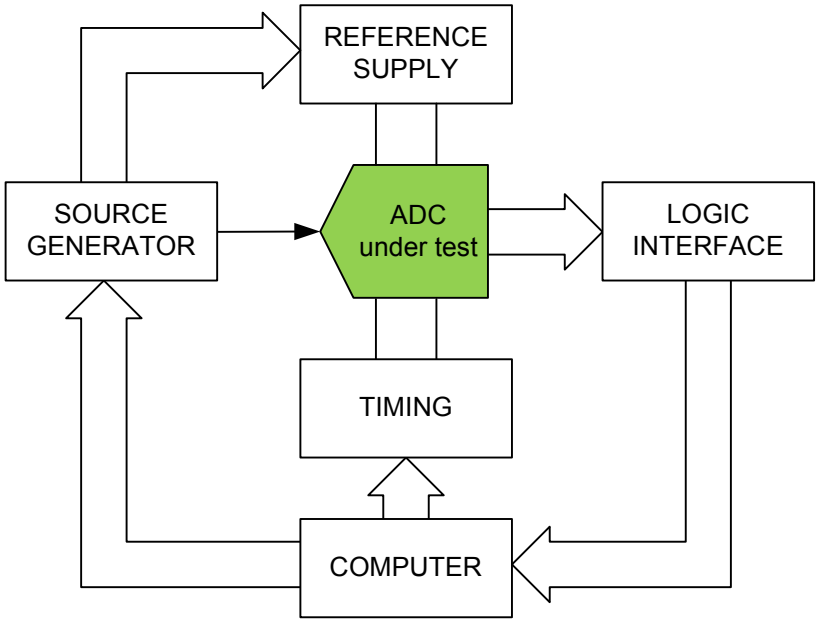


Figure 21: Basic setup for testing of ADCs.

The basic setup uses a source generator, which applies an input stimulus to the ADC under test. The output codes are observed and processed with a computer or test logic. The computer can control the source generator, the timing, and, if needed, the reference supply. The source generator must provide the input stimulus with an accuracy of at least 1 LSB better than the ADC under test.

4.4 ADC static characteristic

The ADC static characteristics are related to the parameters which describe the deviations from the ideal transfer function (curve). Figure 22a represents a transfer curve with a variation of quantization intervals, where the interpolating curve is still a straight line. In the second case (Figure 22b) the transfer curve is such that the variation of the quantization intervals leads to a distorted response.

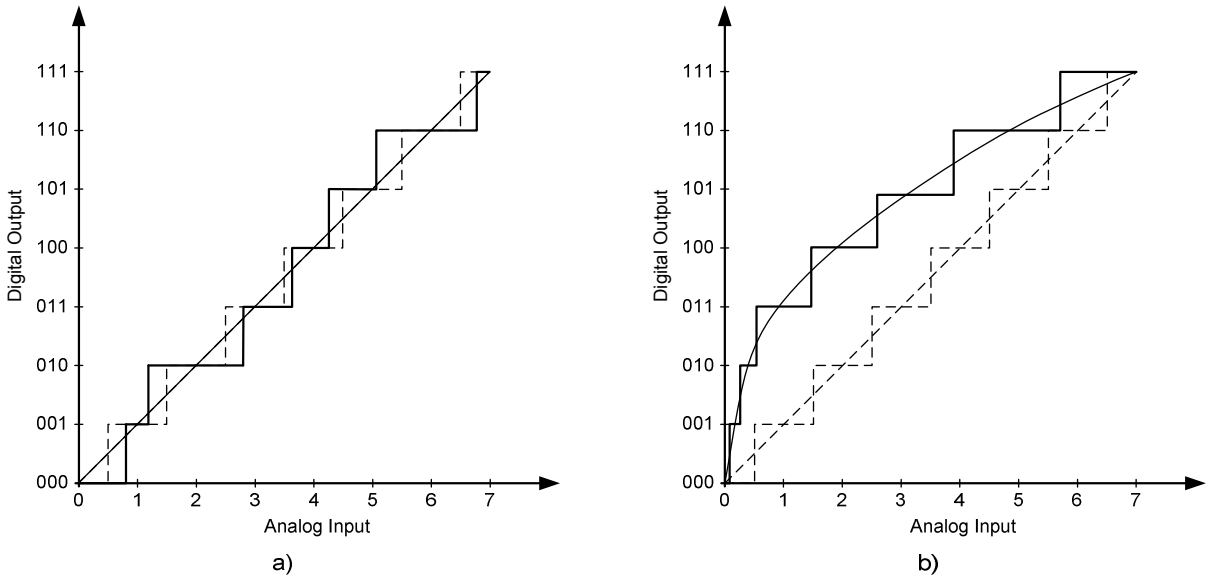


Figure 22: Input-output transfer functions of a real ADC.

The static transfer characteristics in which the deviations of the transfer curve of a real ADC from the ideal are manifested are as follows:

- offset,
- gain,
- differential non-linearity (DNL),
- integral non-linearity (INL),
- total unadjusted error (TUE),
- full-scale error,
- monotonicity,
- hysteresis
- missing code.

The first four listed parameters are the most common static ADC parameters used for describing the behavior of the ADC. They are commonly provided parameters in datasheets. In the following subsections they will be described and explained in more detail.

The total unadjusted error (TUE) is the worst-case deviation from the ideal ADC performance. It is an all-inclusive specification that comprehends the offset error, the gain error and the linearity errors.

The full-scale error is the deviation of the actual full-scale transition point (last code transition) from the ideal value. It is usually measured in LSB. The full-scale error presents an offset error and a gain error added together.

Monotonicity is a feature that produces output codes consistently increasing with the increasing input signal and consistently decreasing when the input signal is decreasing. In other words, the output code remains constant when the input is not changing, or changes consequently with the same direction as the input.

Hysteresis is the difference between the code transitions depending on the direction of the input signal. In the ideal case the particular code transition occurs at the same voltage input level, regardless of the direction of input. If this is not the case the hysteresis is the maximum of such differences.

The missing code is a digital code, which never appears at the ADC output. The missing code cannot be reached by any input value. If that happens, the code width of such a code is zero.

4.4.1 Offset–offset error

The offset describes a shift of the transfer characteristic. The offset moves the transfer characteristic in such a way that all the threshold voltages are shifted by the ADC offset. While the offset in general affects all the codes, it is defined with the first code transition. The first output code transition should occur at an analog voltage that is half of the quantization interval (0.5) LSB above zero. The offset error is the deviation of the first actual code transition from the ideal, which occurs at 0.5 LSB, as depicted in Figure 23.

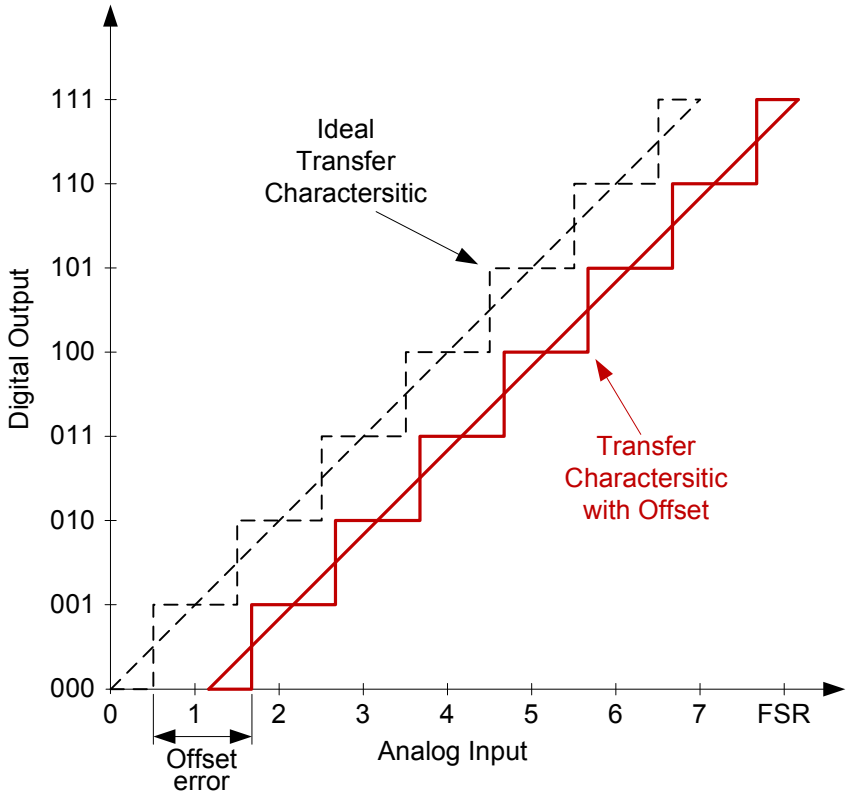


Figure 23: Transfer characteristic with offset and offset error.

The offset error may have a positive or negative value, when the first transition code transition occurs higher or lower than ideal, respectively. The offset error can be expressed in LSB, volts or as a percentage of the full-scale range.

4.4.2 Gain–gain error

As already mentioned, the gain of the ideal linear ADC is in general considered to be unity, equation (7). Consequently, the slope of the code center line is equal to 1. The gain error defines the deviation of the slope of the actual ADC code center line from the ideal (Figure 24).

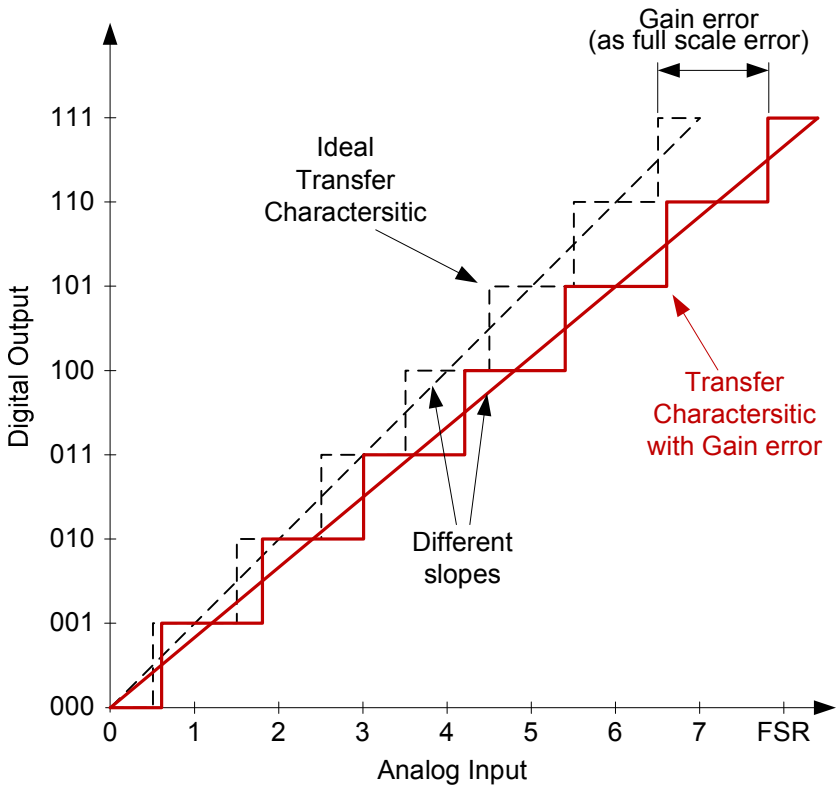


Figure 24: Transfer characteristic with gain error.

The gain error can be measured as the difference between the ideal last code transition and the actual (real) last code transition. When performing a measurement in such a manner, it must be considered that the offset error also influences the result. When using this definition, the gain error is called the full-scale error.

4.4.3 Differential non-linearity

In the case of the ideal linear ADC, the quantization intervals (step size) are equal. This is called the nominal code width (Q_N), defined with equation (8). The nominal code width is a voltage equivalent to 1 LSB. The code width $Q(i)$ is defined as the difference between two consecutive threshold voltages V_{TH}:

$$Q(i) = V_{TH}(i+1) - V_{TH}(i)\,. \tag{10}$$

In other words, the code width is the range of the input values that produce the same digital output code.

The DNL measures the deviation of each code width $Q(i)$ from nominal code width, which is 1 LSB. The DNL error for the code i is defined as the relative difference of the actual code width and the nominal code width:

$$DNL(i) = \frac{Q(i) - Q_N}{Q_N}\,. \tag{11}$$

The DNL describes a step error and can differ from code to code, as shown in Figure 25. The DNL is sometimes referred to as the differential linearity error (DLE).

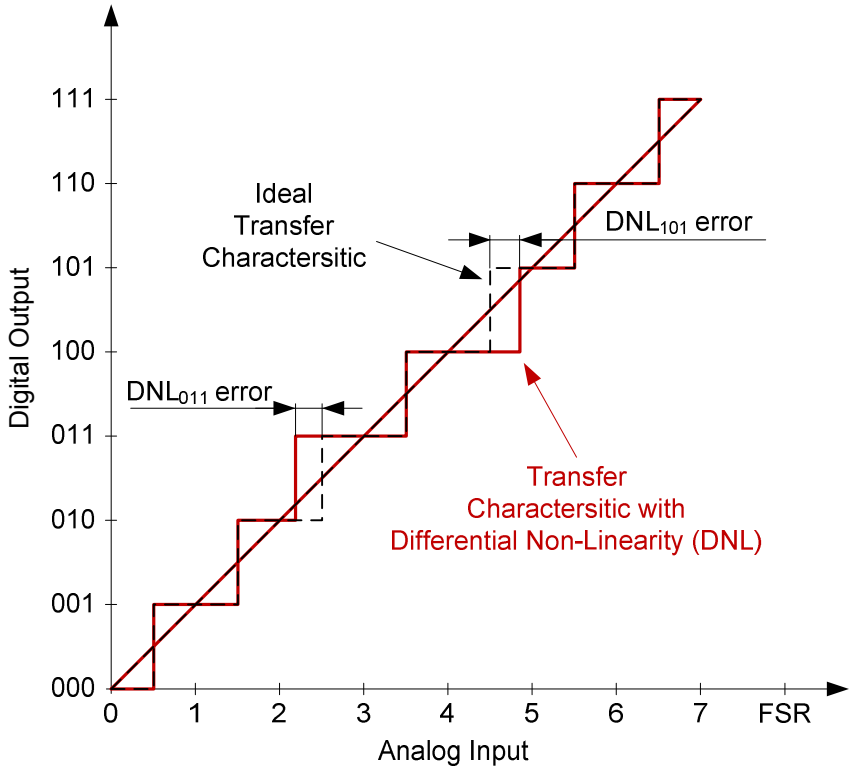


Figure 25: Transfer characteristic with differential non-linearity (DNL).

The DNL value is given in LSB. In the case of a positive DNL, the input voltage range that produces the same code is wider than nominal, and narrower in the case of the negative DNL, respectively.

The maximum DNL is the maximum value of $|DNL(i)|$, considering all the non-extreme codes. Non-extreme code stands for the codes with both lower and upper transition points. In the case of an extreme code it is not possible to define the code widths and consequently it is also not possible to determine the DNL.

The maximum DNL is often referred to as simply the DNL, especially in the datasheets provided by ADC manufacturers. Additionally, the root mean square RMS of the DNL can be given as:

$$DNL_{RMS} = \sqrt{\frac{1}{2^n - 2}\sum_{1}^{2^2-2}\left(DNL(i)\right)^2}\ . \tag{12}$$

4.4.4 Integral non-linearity

The integral non-linearity (INL) determines the deviation of each output code center from the ideal straight center line, as shown in Figure 26. The INL is also referred to as the integral linearity error (ILE), or simply as the linearity error (LE).

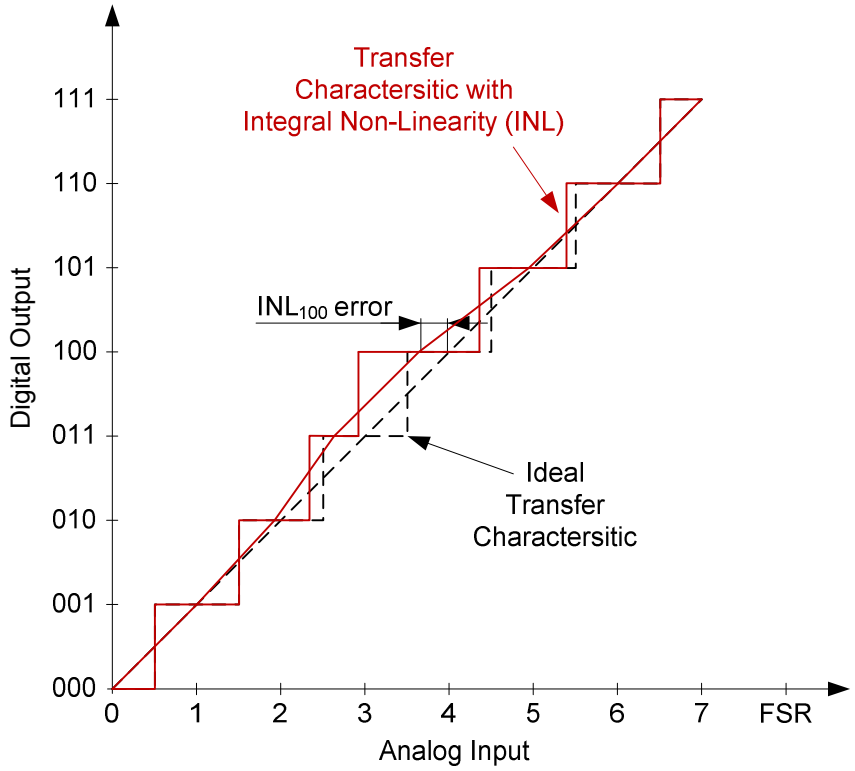


Figure 26: Transfer characteristic with integral non-linearity (INL).

The INL measures the straightness of the transfer function and it can be greater than the DNL. The INL depends on the size and the distribution of the DNL. The INL for code *i* is computed from the cumulative sum of the DNLs of the previous codes as:

$$INL(i) = \sum_{j=1}^{i} DNL(j). \tag{13}$$

There are two ways to measure the INL. The first is called an end-point measurement, and the second a best-fit measurement. The end-point measurement puts a straight line between the end points, as presented in Figure 27a. The best-fit method places a best-fit straight line with minimum deviations (Figure 27b). The end-fit is chosen as the standard since it is more informative and indicates the worst-case INL.

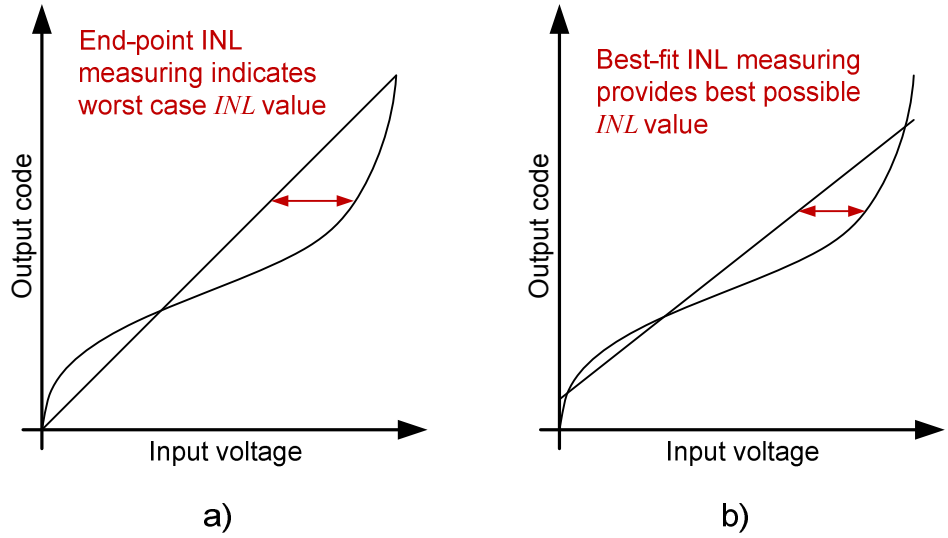


Figure 27: End-point INL measure a) and best-fit INL measure.

Like in the case of the DNL, the maximum value of the INL is usually provided with the ADC specification sheet, indicating the maximum of $|INL(i)|$ when considering all the non-extreme codes.

4.5 Testing ADC static characteristics

One way to verify the ADC static characteristics is to determine the transfer function, by applying numbers of DC voltage steps to the ADC input. For each step the data pairs of the associated input voltage and the output code have to be recorded. By trimming the input voltages, the threshold voltage (V_{TH}) is found when half of the output results fall in one code and the other half in the next one. This is the so-called step-search method. Code-edge testing with a step search can be time consuming, especially when a number of steps have to be applied to obtain the desired resolution.

In a static characteristic test the use of continuous signals is more favorable. In general, the methods with a continuous input stimulus can be divided into the feedback-loop methods and the histogram methods. According to the IEEE Standard for terminology and test methods for analog-to-digital converters [42], three methods are widely in use today: the feedback-loop method, the histogram method with ramp input wave, and the sine-wave histogram method.

The above-mentioned methods can be classified as code-edge test methods, because they measure the code-edge transitions from which all the static characteristics can be obtained. A very similar method to the feedback-loop method is the oscillation-based test (OBT), proposed in [38]. The oscillation-based ADC test, in the same way as the feedback-loop method, forces the input stimulus to oscillate around a desired code transition, but instead of measuring the average oscillation voltage, it measures the oscillation frequency. The only weakness of such a test approach for the ADC is that from the measured oscillation only the conversion time, the DNL and the INL can be calculated.

4.5.1 Feedback-loop testing

As the name indicates, the feedback-loop method uses a feedback loop to control the input stimulus. The method was first introduced in 1975 by [39]. There are several adaptations of this technique [40, 41], also referred to as the servo or servo-loop method. The feedback loop method is also included and described in IEEE Std 1241 [42].

A digital comparator monitors the output code from the ADC. On the test source side an analog integrator is employed as the analog voltage generator. The digital comparator controls the slope of the input signal ramp, depending on the comparison, as shown in Figure 28a. When the ADC output code changes from a reference code the integrator changes the input slope, causing the input voltage to oscillate, as shown in Figure 29. Because of the ADC conversion time and the loop delay time the input signal experiences voltage overshooting. Measuring the average voltage generated at the ADC input yields the value of the particular code threshold voltage $V_{TH}(i)$. Instead of the analog integrator input stimulus with the DAC at the ADC the input can be applied, as depicted in Figure 28b. As in the case of the analog integrator, once the code transition has been reached, the feedback loop causes the signal to oscillate around the chosen transition. The ADC input for the corresponding threshold voltage is then calculated from the known transfer function of the DAC or is measured with a voltmeter (optional).

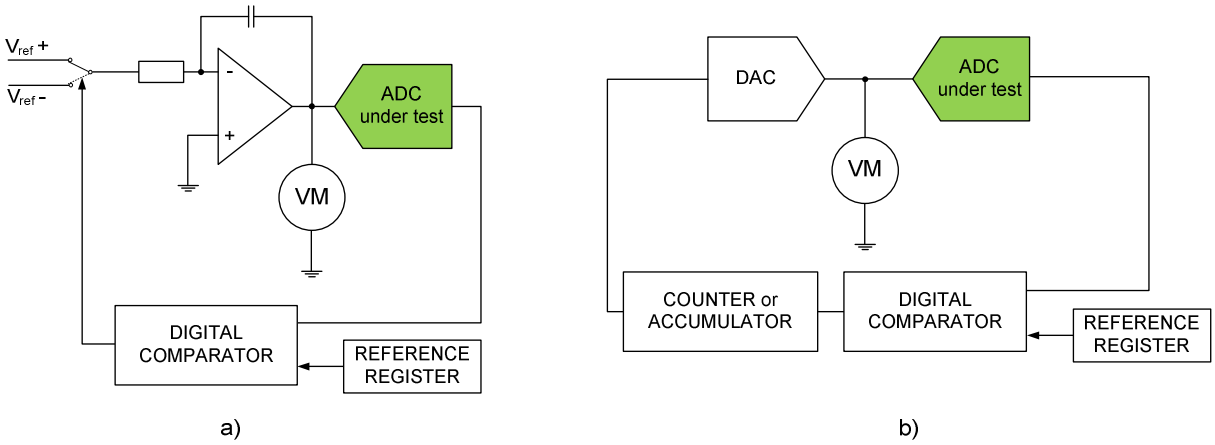


Figure 28: Feedback-loop method with a) analog integrator and with b) DAC as input generator.

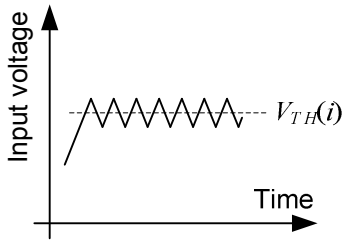


Figure 29: Feedback loop input voltage oscillation.

The feedback-loop test can be easily automated by measuring the threshold voltages of all the ADC codes. The test time can present a major restriction when applying the feedback-loop method. In order to shorten the test times, the test can be performed only for a set of the most significant bits. For feedback loop ADC testing the input stimulus accuracy is also important; in particular, the positive and negative slope rates must be equal when having the analog integrator at the ADC input. Any mismatch of slopes leads to a different ratio of code occurrences at either side of the voltage threshold. This is similar to the case of the DAC, where the step size and the DAC resolution affect the feedback-loop dynamics and, respectively, the accuracy.

4.5.2 Histogram-based test

A histogram is a bar graph that shows how frequently data occur within certain intervals or ranges. The height of each bar in the graph gives the data occurrences (frequency) in the respective interval.

An ADC histogram presents the number of occurrences (code counts) of each output code. For a known periodic input stimulus the histogram of an ideal n-bit ADC can be calculated. In histogram testing the code-transition levels are not directly measured, but derived from the frequency distributions of the ADC codes. With the comparison of the number of code occurrences of an ideal ADC and the actual measured values the static characteristics can be easily identified. However, if needed, the code-transition levels can also be determined.

There are two most common types of histograms, distinguished in terms of the shape of the applied signal at the ADC input:

- ramp histogram,
- sine-wave histogram.

The ramp histogram computed for a linear signal with a uniform distribution (typically, a triangular waveform signal is used) is depicted in Figure 30. The input signal ramps linearly between the extremes of the full-scale range of the ADC.

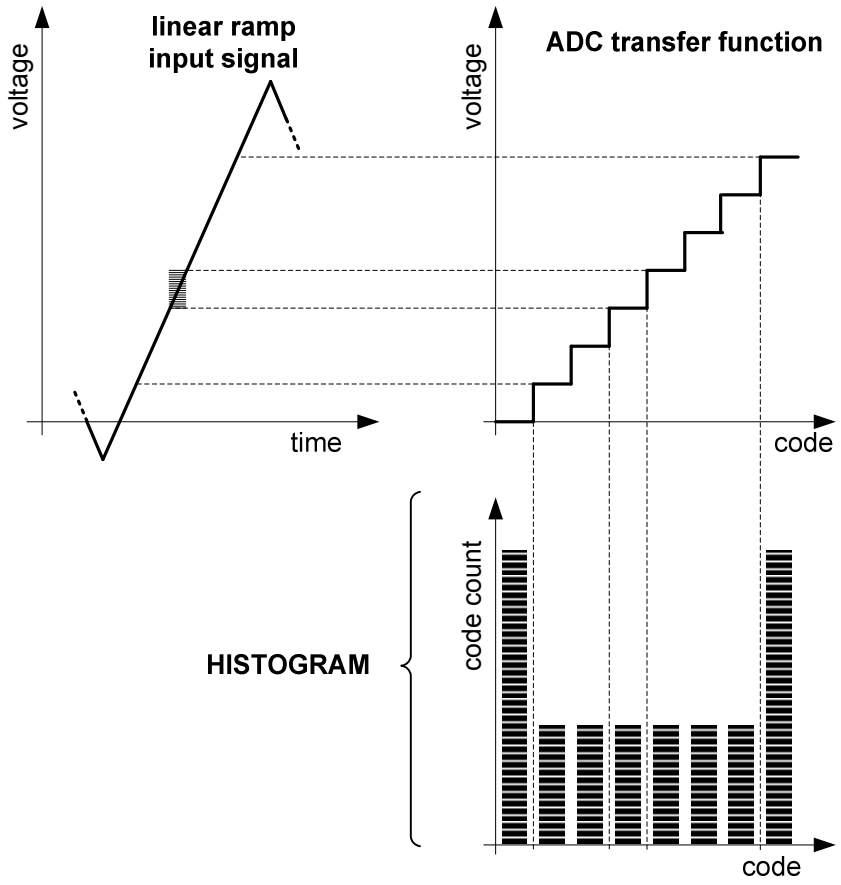


Figure 30: The histogram of ADC code occurrences of ramp input stimulus.

For ramp histograms the ideal values of the code occurrences are equal for extreme and non-extreme codes. After a sufficiently large number of samples the collected ramp histogram provides an accurate measure of the static characteristics. A missing code is easily identified if the corresponding number of occurrences for a particular code is equal to zero. The offset reflects as an increase or decrease of the code counts of the extreme codes. The threshold voltage of a particular code *i* can be calculated from the accumulated code count up to that code [42]:

$$V_{TH}(i) = V_0 + A \cdot \sum_{j=0}^{i-1} H(j). \tag{14}$$

where V_0 stands for the offset factor and the A for the gain factor.

The ramp histogram test method accuracy is restricted by the quality of the input signal and the number of collected samples in the individual code bin. Increasing the number of samples and consequently increasing the number of histogram occurrences, decreases the uncertainty. The quality of the signal refers to the nonlinearity and the noise on the ramp signal [42]. The nonlinearity and noise of the input ramp would produce the errors in the code-transition levels. The uncertainty therefore depends on the standard deviation of the noise σ of the stimulus and the average number of code occurrences (H_{avg}). The error of the ramp histogram method is approximated with:

$$\varepsilon \approx \sqrt{\frac{\sigma}{H_{avg}}}. \tag{15}$$

The ramp histogram test method is normally used when static characteristics need to be measured. The basic difference between the classical measurements of the static characteristics and the histogram approach is that the latter records only the number of times each count appears, while the classical measurements process and calculate the read-out code. This means that there is less data handling, especially in the case of the ramp input stimulus where the histogram in ideal circumstances is characterized only by two values (the count of the extreme codes and the count of other codes).

The sine-wave distribution is no longer uniform as in the case of the ramp signal. The histogram values change dynamically as a sine wave; this is also the reason why a sine-wave histogram is often referred to as a dynamic histogram [8]. The frequency of the sine wave should be chosen according to the ADC sampling frequency. If the test frequency is low enough the dynamic errors will not arise, but the static errors could be obtained. If the frequency of the testing is large enough, some dynamic errors will appear in the results, while others will be averaged out [42].

For sine-wave histograms, as depicted in Figure 31, ADC code occurrences cannot be presented with only two values, as in the case of the ramp signal.

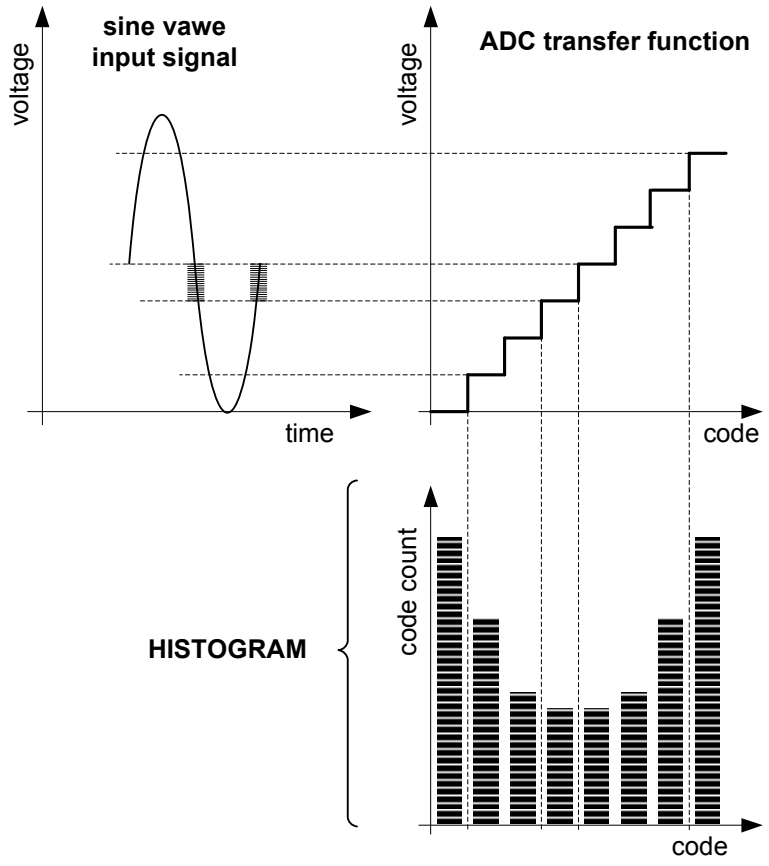


Figure 31: The histogram of ADC code occurrences of sine-wave input stimulus.

The main attention, for either type of histogram, must be paid to the input stimulus accuracy, because it has a direct impact on the test results. The ramp histogram test advantage lies in the lower number of samples required due to the constant ideal code counts. Data processing in this case operates only with two ideal values of code count and the actual measured values. On the other hand, a sine-wave signal generation of high frequency and accuracy can be more easily achieved.

4.5.3 Oscillation based test

In the oscillation-based test (OBT) method proposed in [38], a CUT is transformed into an oscillator. The technique modifies the CUT in such a way that oscillation is induced. Arabi and Kaminska proposed different mechanisms for converting CUT to an oscillator. The most effective one is a closed-loop system [43] which employs the feedback loop, including a transfer function F_{loop} and an adder, as shown in Figure 32.

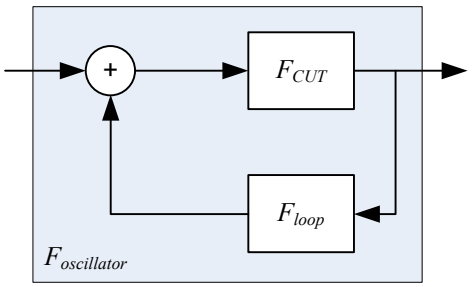


Figure 32: Converting a CUT in an oscillator.

The proposed approach can be applied to any analog or mixed-signal circuit. The OBT approach can also be easily extended to BIST. In some literature [10, 44] a built-in self-test based on OBT is simply called the oscillation built-in self-test (OBIST). The BIST concept based on OBT first divides complex systems such as SoC to simpler individual functional circuits [10, 38, 44-49]. During the test each of these CUTs is converted to an oscillator. The CUT produces a test output signal whose oscillation frequency is inherently related to the fault-free structure of a particular CUT. The oscillation frequency within a tolerance margin is considered as a test parameter in such a way that it is compared to a reference value, analytically determined for ideal circumstances or obtained on a known-good circuit operating under the same measurement conditions. The discrepancy between the oscillation frequency of a CUT and the reference value indicates possible faults. The OBT structure consists of the logic control structure for accessing to the individual CUT and the block for a comparison of the reference and the measured oscillation frequency. In the case of an on-chip determination of the test result the control logic usually employs a frequency-to-number converter for easy processing. In the case of the off-chip comparison the signal is conveyed to the external test equipment. The basic OBIST structure is depicted in Figure 33.

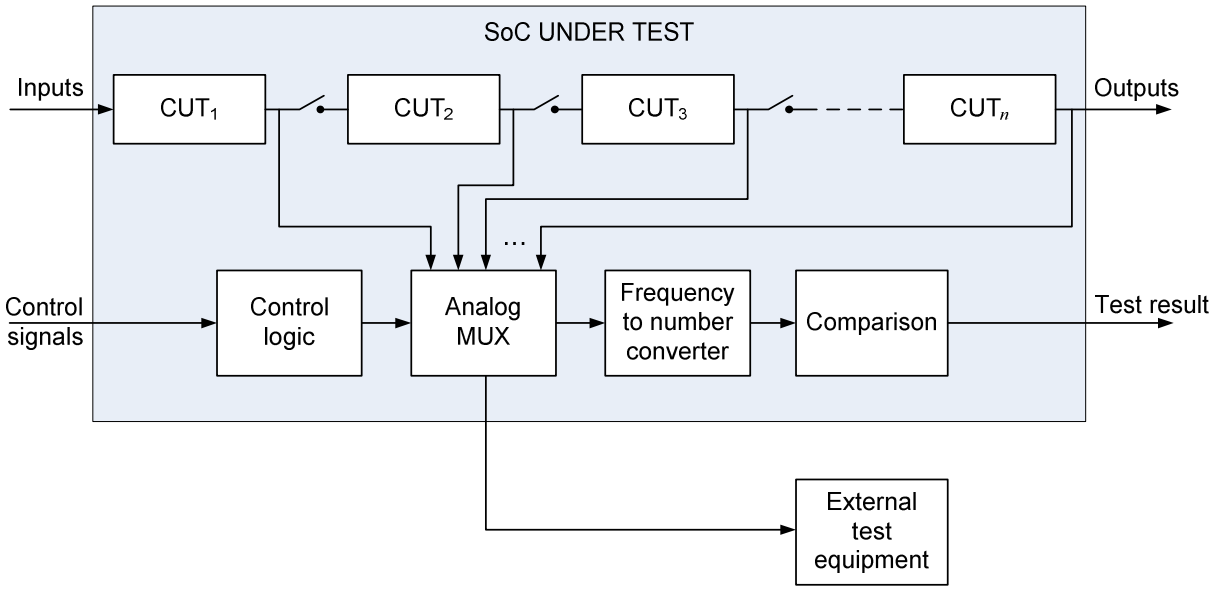


Figure 33: Oscillation built-in self-test structure.

The main advantages of OBT are that no analog (external) stimulus is needed because the CUT is converted into an oscillator and the measuring of the frequency can be a simple measurement with a digital frequency counter. In general there are no other methods to beat the simplicity of the OBT technique. On the other hand, the OBT is hard to adapt for diagnostic purposes, especially in the case when the fault prevents the oscillation. The OBT relies on the fault model accuracy to assess the fault coverage and cannot test for all of the parameters related to the specification (for example, noise).

The OBT technique has been applied, not only to the ADC, but also to the different kinds of mixed-signal circuits [44, 46, 50-54]. In OBT for the ADC the setup of the test is similar to the feedback-loop method employing the feedback loop and analog integrator. The main difference is that in the case of OBT oscillation frequency is measured instead of the average voltage at the input of the ADC. In the OBT of (ADC), a feedback loop is provided, which forces the ADC to oscillate around a selected code *i* [10]. The feedback loop is fed to a circuit that generates a triangle wave signal of symmetrical slope. Figure 34 shows the basic OBT of ADC measurement set up with two different integrator circuits.

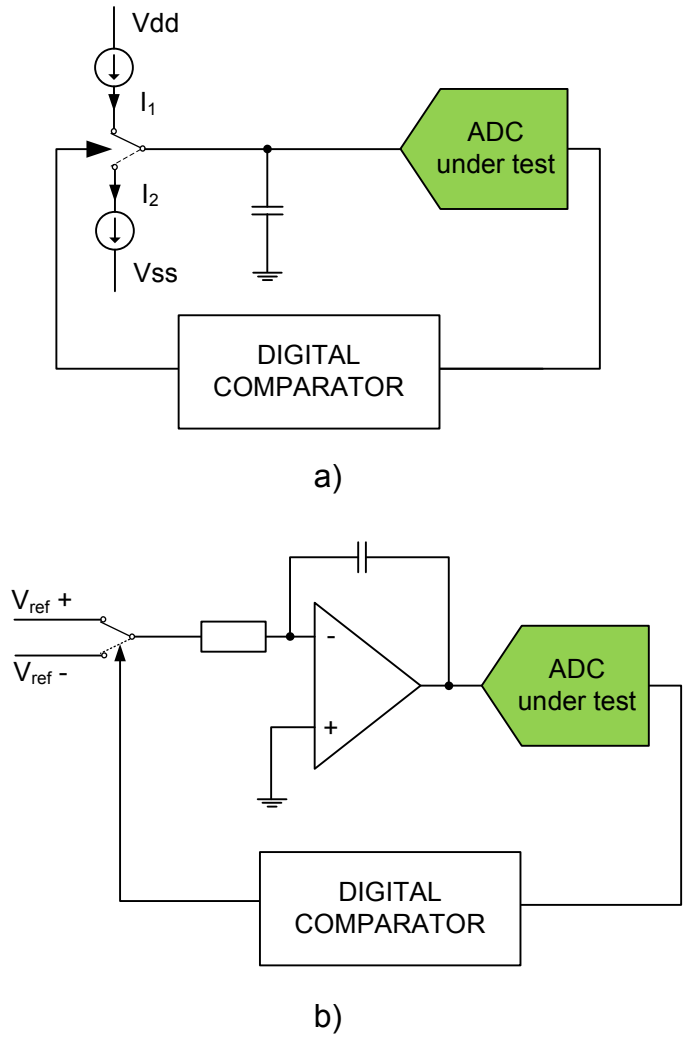


Figure 34: OBT structures of ADC, a) integrator circuit with current generator, b) integrator circuit with operational amplifier.

With the described OBT technique we can only measure the non-linearity of the ADC. The DNL and INL are calculated from the code width, which is obtained by measuring the oscillation frequency oscillating around the desired code *i* (or code *i* transition voltages), as presented in Figure 35.

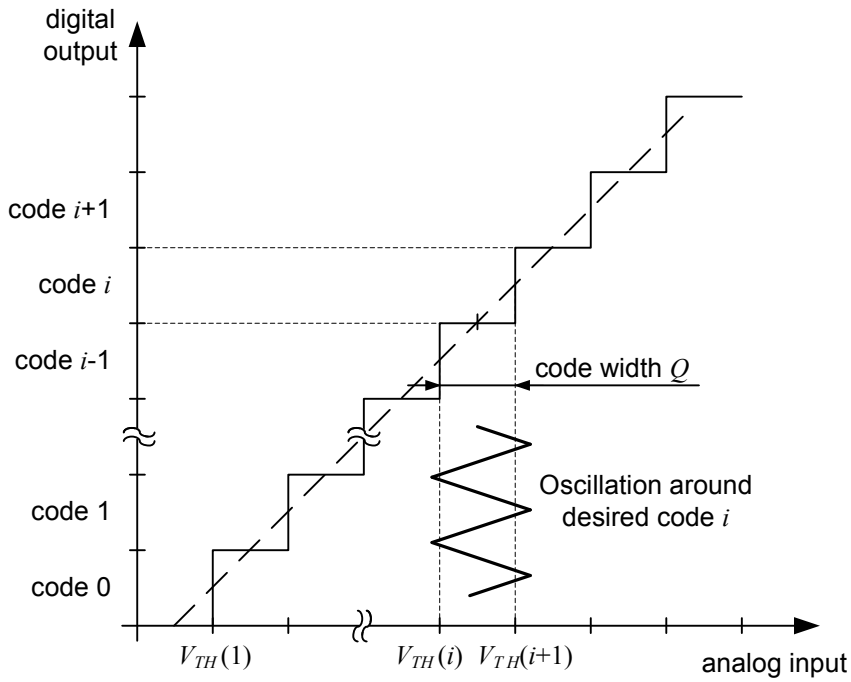


Figure 35: Oscillation around specified code.

Evaluation of code width from the measured oscillation frequency represents the basis of the OBT test method of ADC [10].

5 Histogram test for embedded ADC cores

The classical measurement of the static characteristics of an ADC involves the generation of stimuli (sine-wave or triangle-wave) and the read-out of the generated code processed by automated-test equipment (ATE). In the case of the ADC core implemented in a system-on-chip (SoC), as shown in Figure 36, the BIST approach is often a preferred solution due to a potential communication bottleneck between the ATE and the tested ADC core.

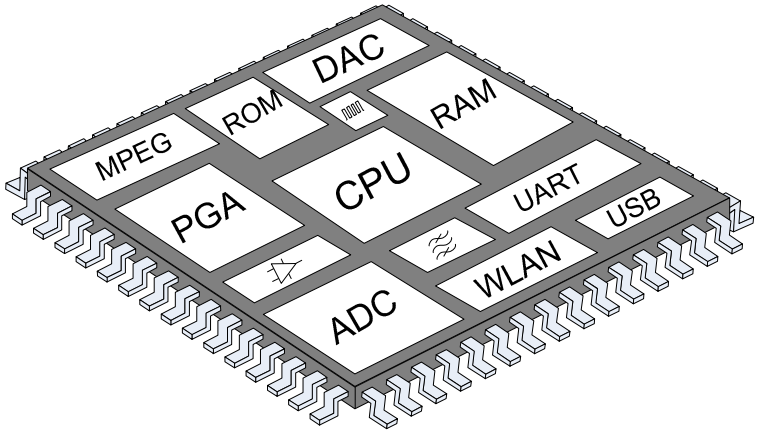


Figure 36: Implemented ADC and other digital and mixed-signal cores in a SoC.

However, since it may be hard to generate reference stimuli in a SoC, the ADC core BIST is often reduced only to processing and evaluating the ADC outputs, while the ADC input is connected to an external stimulus generator.

The histogram method is an established approach for determining the static parameters. The generation of the stimuli is similar to the classic measurement; however, the method only records the number of times each code appeared (which can be viewed as a histogram).

While the theoretical background for the histogram-based ADC test is well-known [7, 8, 55] and the area overhead for different alternative implementations has been estimated [56, 57], reports of implementations in practice and associated measurements are rare and not publicly available.

The histogram-based test of the ADCs has been in use for more than two decades and different aspects of its application have been discussed by numerous authors. Earlier references [58-61] primarily concentrate on how the code density can be interpreted to compute the differential and integral nonlinearities, gain error and offset error, and estimate the achieved accuracy under different measurement conditions. More recently, the accuracy of ADC testing with sine-wave stimulus has been analyzed under the assumption of quasi-coherent sampling and a bound on the variance of the transition level estimators have been derived in [62]. The error in the estimation of the transition voltages in the presence of additive noise was analyzed in [63]. As a continuation of this work, the error in the code bin widths has been studied, the error of the differential nonlinearity estimation has been analyzed and expressions for the amount of overdrive to minimize the error have been proposed in [64]. Another paper by the same authors [65] analyzes the precision of the estimates of the ADC gain and offset error that are obtained with the histogram method. An improved histogram-based approach, which also reveals dynamic performance, such as the effective number of

bits, is proposed in [66]. A novel approach based on small-amplitude waves is proposed in [67] and further revised in [68].

Papers [56, 70, 71] focus on the implementation of the histogram-based test of ADCs in a BIST arrangement. In general, a complete BIST scheme requires the definition of a reference analog input generator and a digital output response analyzer. As on-chip generation of the reference stimulus can be regarded as a classical problem with known solutions [72], the authors focus on defining the digital output response analyzer. They employ a triangular input waveform and derive the procedures for the computation of the offset, gain, DNL and INL. The proposed BIST logic that implements the above procedures is verified with simulations for different lengths of ADCs. In [73] some improvements to the above BIST logic are proposed by performing DNL, INL, offset and gain error calculation procedures in parallel.

So far, most of the proposed solutions have been evaluated by simulations. Their application in SoC testing in practice is still an open issue. Paper [74] can be regarded as an attempt at an implementation in a test wrapper conforming to IEEE Std 1500 [6]. However, the paper describes a general mixed-signal test wrapper design and does not provide any details of possible implementations of test techniques in practice. The lack of reported experimental evidence encouraged our work on the design of a histogram-based test technique in a IEEE Std 1500 wrapper together with a thorough evaluation of the implemented test infrastructure on experimental case studies. We implemented three extreme versions (sequential, RAM-based and processor-based methods) of the histogram-based BIST in a IEEE Std 1500 wrapper regarding test time/hardware overhead trade-off, all evaluated by laboratory measurements. In order to provide realistic measurement conditions we introduced external stimuli via the analog boundary module of a IEEE Std 1149.4 compliant experimental test chip [75].

In our implementation (Figure 37), an off-the-shelf ADC (MAX165) was chosen as the unit-under-test. This is an 8-bit microprocessor compatible ADC. The ADC under-test was connected to a triangle-wave signal source via MOS switches. For this purpose the analog boundary module (ABM) of a 1149.4 test chip implemented in collaboration with LIRMM in the frame of Proteus project [75] was employed. In this way, the BIST configuration at the core input is emulated. The ADC digital responses are processed by a Spartan3 XC3S200 FPGA. The measurements were carried out to evaluate the performance of the proposed test solution.

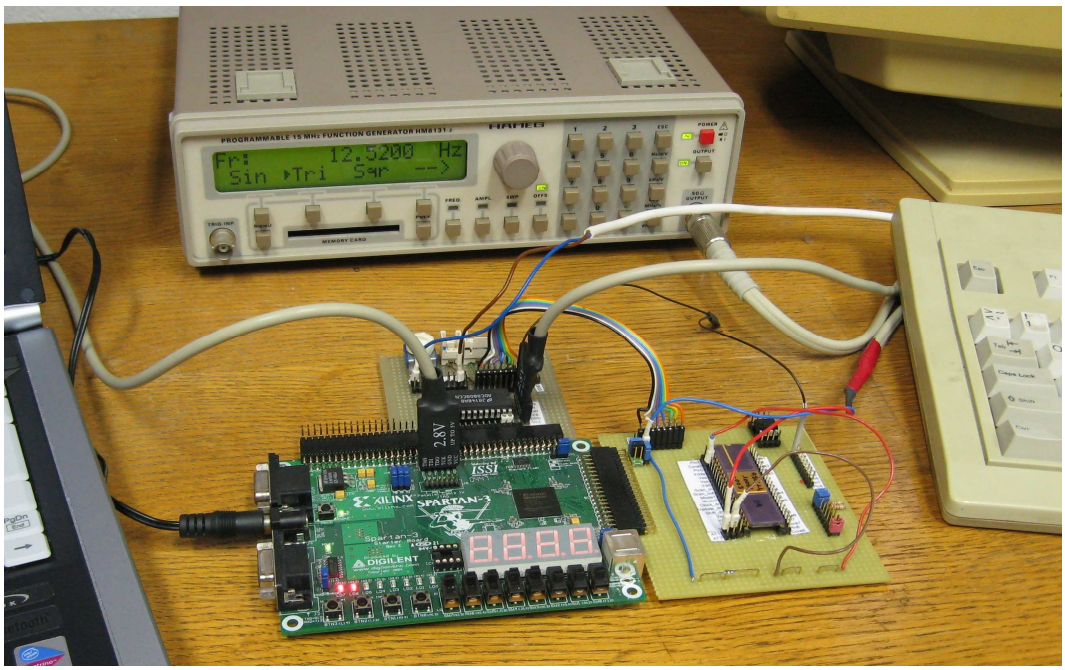


Figure 37: Laboratory prototype for performing measurements.

The main improvements of our implementation are in providing the coherent sampling of the input stimulus and in accomplishing more realistic measurement conditions, taking into account the impact of MOS switches in the implemented test wrapper. The implemented histogram methods have been validated by employing a digital counter that simulated the ideal ADC output.

The histogram test technique is based on a known amplitude distribution. The histogram represents the counts of each ADC code (i.e., amplitude) for the applied input signal. The ideal histogram for the specific stimulus waveform is characterized by an analytically determined ideal count (H_{ideal}) for each code. A triangle input signal is the most convenient for the static characteristics BIST, because the ideal ADC static performance characteristics are then characterized only by two values (the count of extreme codes and the count of other codes).

5.1 Offset error calculation

The first output code transition should occur at an analog voltage that is 0.5 LSB above zero. The offset error is the deviation of the first actual code transition from 0.5 LSB. The offset error (ε_{offset}) determined with histogram-based test is calculated using the following expression:

$$\varepsilon_{offset} = \frac{H(2^n - 1) - H(0)}{2 \cdot H_{ideal}} \tag{16}$$

where H_{ideal} is the ideal number of code counts, $H(2^n\text{-}1)$ is the number of measured occurrences of the MSB code and $H(0)$ of the LSB code, respectively. The influence of the offset error is shown in Figure 38.

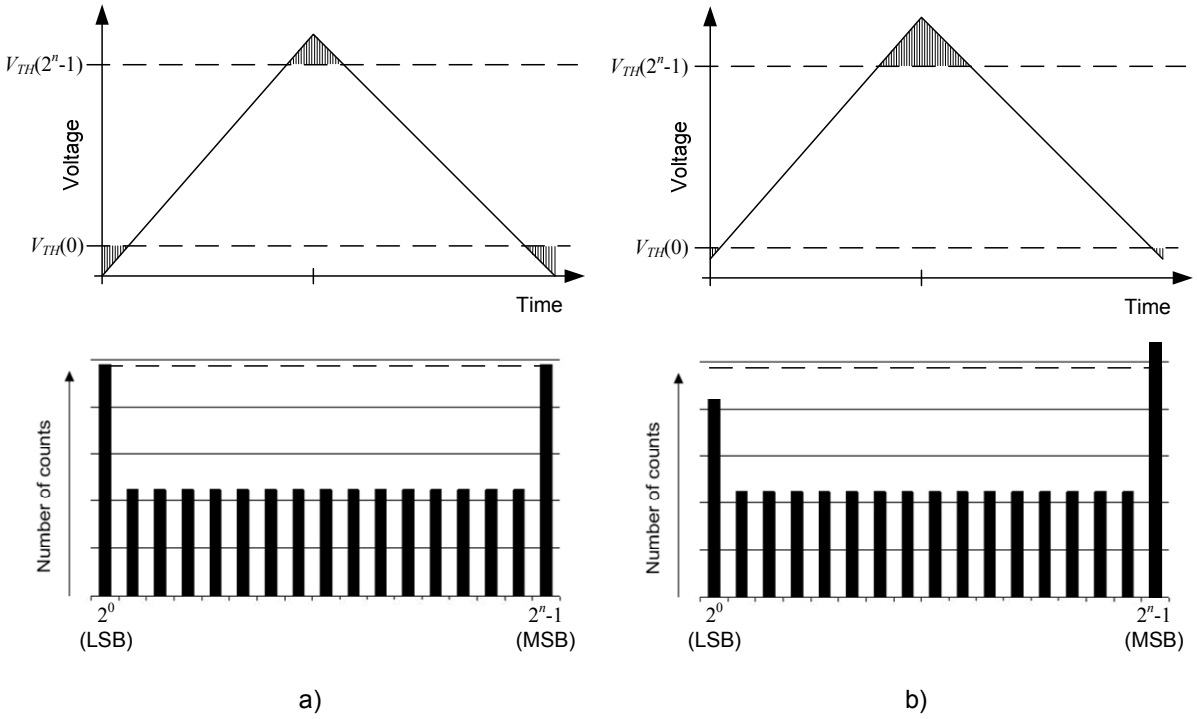


Figure 38: Histogram of an ideal ADC a), histogram of an ADC with offset error b).

5.2 Gain error calculation

The gain error is the deviation of the actual level at the last transition from the ideal level. The ratio between the count (H) of any non-extreme code and the ideal count (H_{ideal}) determines the ADC gain at that code. While the measured value varies from code to code the ADC gain-error (ε_{gain}) estimation in the histogram test is determined by averaging the gain over m central codes of the ADC range. The gain error is shown in Figure 39 and calculated from:

$$\varepsilon_{gain} = \frac{\sum_{i=N_1}^{i=N_2} H(i)}{m \cdot H_{ideal}}. \tag{17}$$

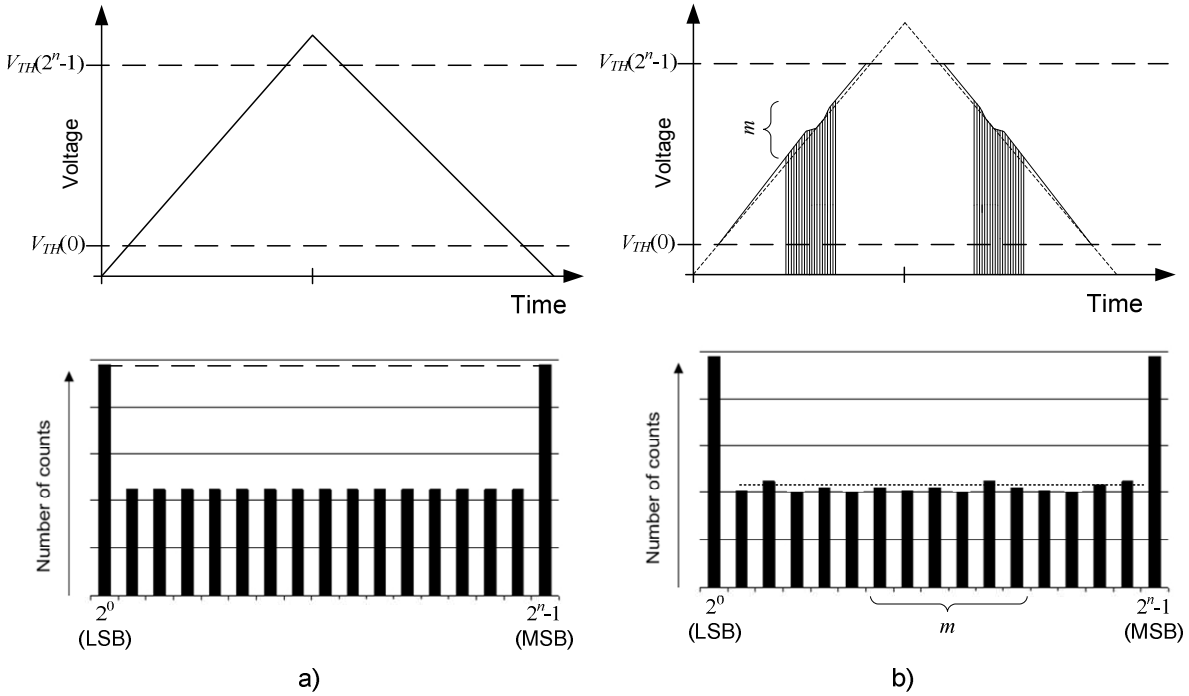


Figure 39: Histogram of an ideal ADC a), histogram of an ADC with gain error b).

5.3 Non-linearity error calculation

DNL measures the code width of each step and its deviation from the ideal value (1 LSB). Given the histogram data, DNL is defined as the relative difference between the measured and the ideal counts

$$DNL(i) = \frac{H(i) - H_{ideal}}{H_{ideal}}. \tag{18}$$

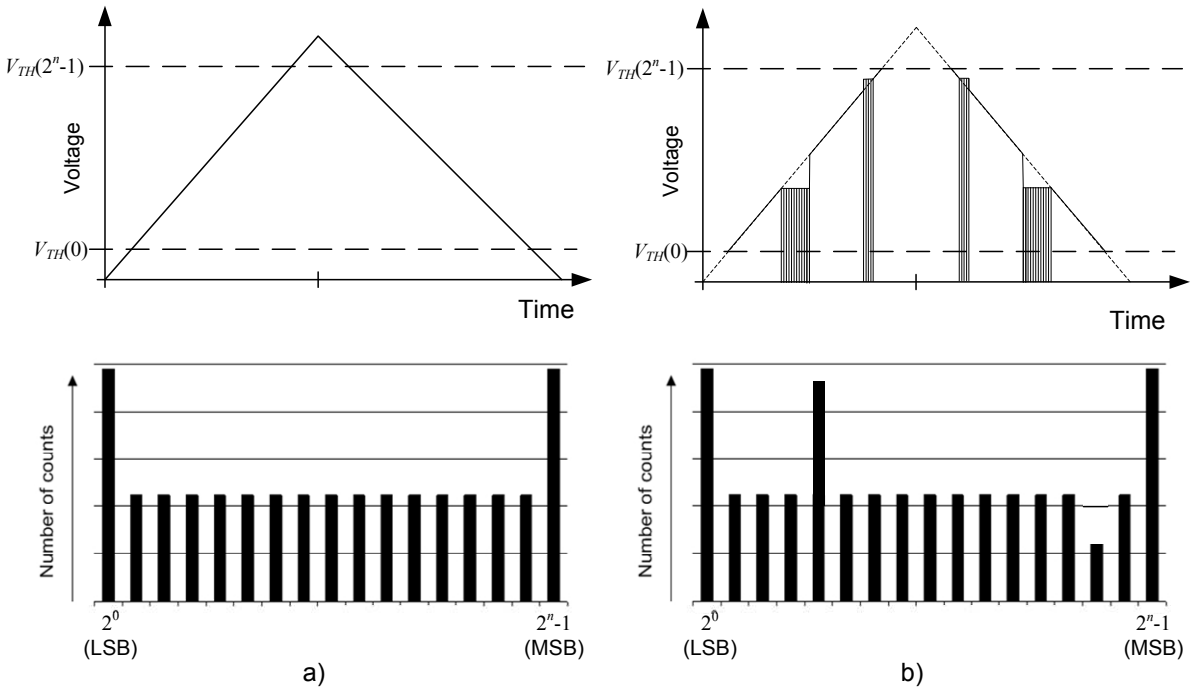


Figure 40: Histogram of an ideal ADC a), histogram of an ADC with DNL error b).

The INL determines the deviation of each output code center from the ideal straight center line. The INL for code *i* is computed from the cumulative sum of *DNL*s of the previous codes.

$$INL(i) = \sum_{j=1}^{i} DNL(j). \tag{19}$$

The above equations include the operations of addition, subtraction, and division. While addition and subtraction are fairly simple operations, the hardware implementation of division is more demanding. However, if the denominator is the power of 2, the division can be performed by shifting the decimal point. In our case the denominators are H_{ideal}, $2 \cdot H_{ideal}$, and $m \cdot H_{ideal}$. In order to simplify the division, the H_{ideal} and m should be the power of 2.

5.4 Histogram BIST

For ADC BIST implementation based on a histogram is, as described in [56, 70, 71], most convenient a triangle wave input stimulus. In the case of the triangle-wave input signal the histogram is characterized only by two values (the count of extreme codes and the count of other codes). The count of extreme codes is normally greater than the count of the non-extreme codes due to the overflow of the input signal. The principle of histogram testing is sketched in Figure 41.

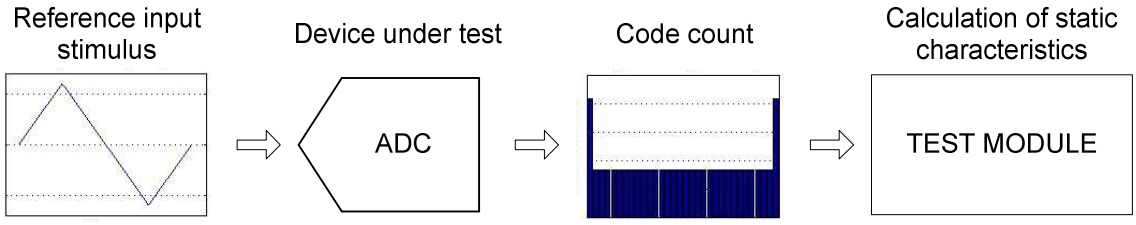


Figure 41: The principle of the histogram test with a triangle-wave input signal.

Coherent sampling must be provided: the ratio between the number of samples *N* and the number of input signal periods *M* must be equal to the ratio between the sampling frequency and the frequency of the input signal. In addition, the slope of the input signal must be

determined in such a way that there is an equal number of occurrences of all non-extreme codes.

The concept of the implemented histogram-based BIST in a IEEE Std 1500 wrapper is shown in Figure 42a. We use a triangle-wave input signal, which in an ideal case results in a constant code count for non-extreme codes and an equal code count for extreme codes.

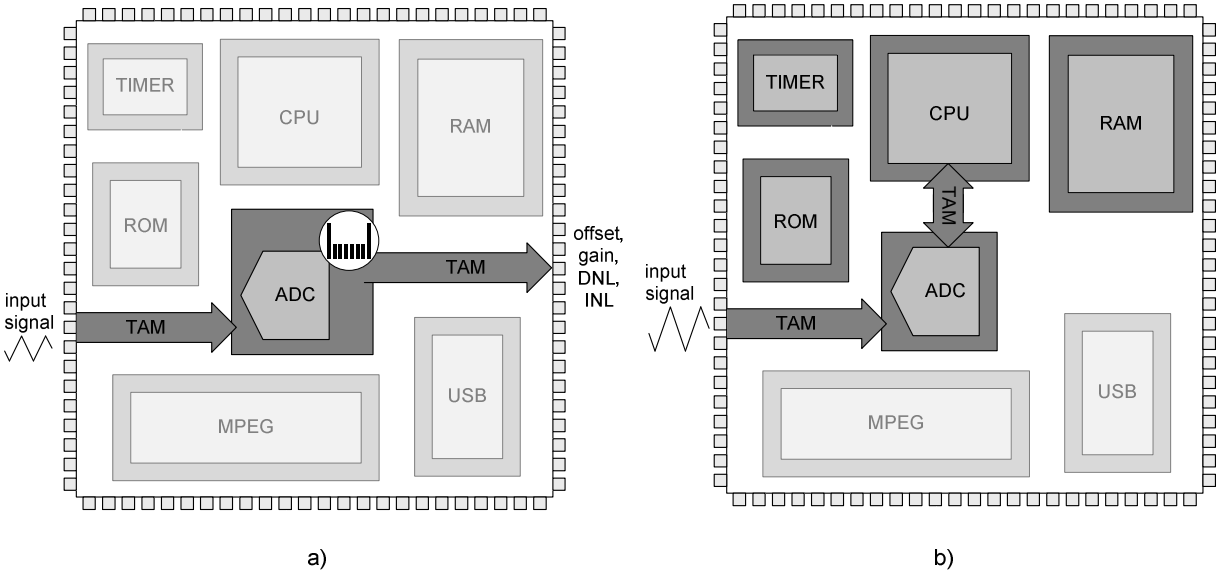


Figure 42: The concept of a) implemented BIST in IEEE Std 1500 test wrapper, b) processor-based BIST.

We first implemented two different BIST structures, targeted either at a low hardware overhead or at a minimum test time. The first solution performs the measurements and computations of the static parameters in a sequence of individual steps. The ADC output data is processed on the fly in each step and the results (i.e., offset, gain, maximum value of *DNL(i)*, maximum value of *INL(i)*) are transmitted out via the test access mechanism. The second solution first collects the complete ADC test responses and stores them in a RAM. Next, the computation of the static parameters is performed and the results are transmitted via TAM, as in the previous case. We denote the first solution as the *sequential BIST* and the second as the *RAM-based BIST*.

In addition, another solution (shown in Figure 42b) was conceived in which a processor core performs the ADC core test and computes the required test parameters.

5.4.1 Sequential BIST

The sequential BIST structure depicted in Figure 43 operates in accordance with the concept proposed in [56]. It was, however, slightly modified in order to completely automate the evaluation of the DNL and INL parameters. The BIST structure is composed of three basic blocks:

- The detector module, which monitors the codes generated by the ADC and the signals when the code is equal to the pre-set value,
- The exploitation module, which receives the signals from the detector module and performs the required operations (counting, complementing, etc.),
- The control module, which coordinates the processing of both modules and connects the BIST structure to the IEEE Std 1500 test infrastructure. The control module is a state machine which generates the corresponding control signals of the detector module and the exploitation module. It also coordinates the operation of the BIST and the IEEE Std 1500 test wrapper logic.

The detector module contains a counter and a comparator. The counter is used for setting the required preset value that is stored in the DM register. The contents of the DM register are compared with the ADC code. If the ADC code equals the contents of the DM register, the exploitation module is triggered. The exploitation module is an up/down counter with a clear. In addition, it is capable of calculating the complement of the current value. The temporary result is stored in the EM register.

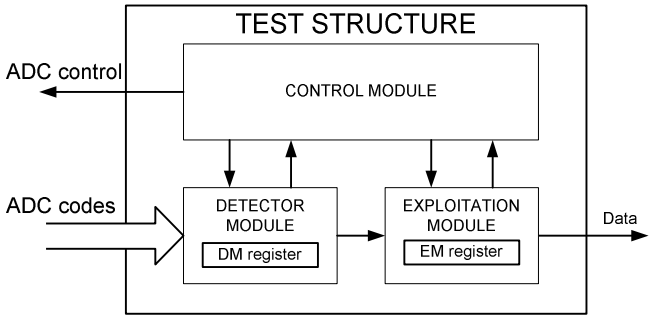


Figure 43: Sequential BIST structure.

The calculations of the offset, gain, DNL and INL are separate processes that are performed in a strictly sequential manner:

Offset calculation. In reference to expression (16), the offset is determined as a difference in the number of occurrences of the two extreme codes divided by $2 \cdot H_{ideal}$. This is achieved by first initializing the EM register to 0 and then increasing its contents when the ADC code is "11...1" and decreasing it when the code is "00...0". Assuming that $H_{ideal} = 2^P$ the division by 2 H_{ideal} is implemented as a (P+1) shift operation. P is an arbitrary integer number; the choice depends on the desired accuracy of the measurements, as shown later. The procedure for the computation of the offset is:

Algorithm 1: The computation procedure for the offset error.

```
initialization: EM=0
for samples i=1 to N {
 if (ADC_LSB=1) {
   DM = 11...1;
   if (ADC = DM) EM++
 }
 else {
   DM = 00...0
   if (ADC = DM) EM--
 }}
if (EM_MSB=1) complement EM
offset = EM >> (P+1) //division performed as shift
```

Gain calculation. The inverse of the gain is determined as the ratio between the average count of m central codes and H_{ideal}. Again, m is selected as a power of 2 ($m = 2^Z$) so that the division is performed as a shift operation. Let us denote the first of the m central codes by N_1 and the last by N_2 ($N_2=N_1+m$-1). The procedure for the computation of the inverse of the gain is:

Algorithm 2: The computation procedure for estimating the gain error.

```
initialization: DM=N1, EM=0
while (DM ≤ N2){
```

```
  for samples i=1 to N{
    if (ADC = DM) EM++
    }
  DM++
}
gain = EM >> (P+Z) // division performed as shift
```

DNL calculation. This calculation is slightly different from the procedure proposed in [56] in order to reduce the number of arithmetic operations. The *DNL* is calculated for each ADC code (except for the two extreme codes). For the selected code, the contents of the EM register are first initialized to -2^P and then increased by the number of occurrences of the code. In this way, the subtraction in expression (18) is eliminated. The highest absolute value of the EM register is taken and divided by H_{ideal}. (If the contents of EM register are negative, its complement is computed.) The procedure for the computation of *DNL* is:

Algorithm 3: The computation procedure of the DNL.

```
initialization: DM=1, DNL=0
while (DM < 2n-1){
  EM = -2P
  for samples i=1 to N {
    if (ADC = DM) EM++
  }
  if (EM_MSB=1) complement EM
  if (DNL < EM) DNL=EM
  DM++
}
DNL = DNL >> P    // division performed as shift
```

INL calculation. For a better understanding we rewrite the expression (19) in the following way

$$INL(0) = 0. \tag{20}$$

$$INL(i) = INL(i-1) + DNL(i). \tag{21}$$

At the beginning the counter is initialized to 0. After accumulating the number of hits for the current code H_{ideal} is subtracted from the contents of the EM register. The highest absolute value of the EM register is taken and divided by H_{ideal}. The procedure for the INL computation is:

Algorithm 4: The computation procedure for the INL.

```
initialization: DM=1, EM=0, INL=0
while (DM < 2^n-1){
  for samples i=1 to N {
    if (ADC = DM) EM++
  }
  EM -= 2^P
  if (EM_MSB=1){
    complement EM
    if (INL < EM) INL=EM
    complement EM
  }
  else if (INL < EM) INL=EM
  DM
```

```
++
}
INL = INL >> P    // division performed as shift
```

While the sequential BIST structure described above requires a low hardware overhead, the test time may become excessive. This is due to the fact that the sequential approach takes N samples for the calculation of the offset, $m \cdot N$ samples for the calculation of the gain, $(2^n\text{-}2) \cdot N$ samples for the calculation of DNL and $(2^n\text{-}2)N$ samples for the calculation of INL. As an alternative, a RAM-based BIST has been developed.

5.4.2 RAM-based BIST

The RAM-based BIST test structure shown in Figure 44 consists of three components, which play a similar role to their counterparts in a sequential BIST. The three components are:

- The RAM module, in which the ADC code count of a complete histogram is stored,
- The computation module, which computes the ADC static characteristics from the values stored in the RAM,
- The control module, which (similar to the sequential BIST) coordinates the processing of both blocks and connects the BIST structure to the IEEE Std 1500 test infrastructure.

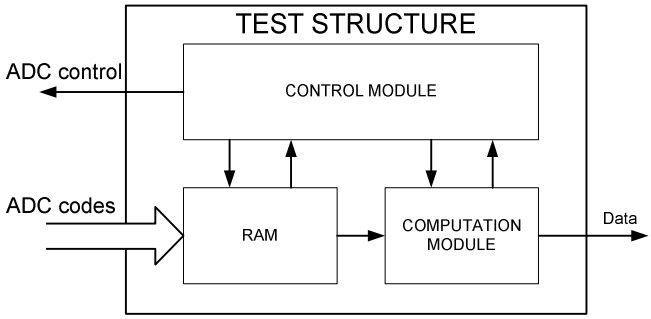


Figure 44: RAM-based BIST structure.

In contrast to the sequential BIST approach, only N samples are taken for the computation of the static parameters of the ADC. The samples are first stored in the RAM, after which the static parameters are computed in accordance with the expressions (16)-(19). In order to eliminate the subtraction in expression (18) the counts are first initialized to -2^P and then increased by the number of occurrences of the code. The RAM is organized in such a way that the ADC code actually presents the RAM address of its code count. In this way, data manipulation is simplified. The computation follows similar principles as before; hence the description in a pseudo-code is omitted.

5.4.3 Processor-based BIST

Processor-based BIST exploits the available microprocessor and other cores on the SoC for the execution of the ADC test. The idea is to reuse the already present resources in the SoC and their interconnections to perform a functional test of the ADC core. The microprocessor with the timer interactions controls the ADC core, gathers the responses and stores them in the SoC RAM. After the stimulation period the gathered responses are evaluated by the processor. In a sense, the processor with a timer performs the role of the control and computation module.

While this approach does not introduce any significant hardware overhead into the digital part of the SoC, additional ROM storage is required for the BIST program storage. In order to

perform the ADC BIST at full speed, which is generally faster than the normal SoC operation, the processor in collaboration with the other cores must be able to respond in time to the incoming ADC codes.

5.4.4 Integrity check of the test structure

The integrity of the test infrastructure must be assured before we can trust the results of the implemented histogram test. For this purpose, the integrity check is performed by replacing the tested ADC with a counter, as shown in Figure 45. The counter generates codes corresponding to the ideal ADC with an ideal triangular input waveform and the resulting static parameters are checked (*offset* = 0, *gain* = 1, *DNL* = 0, *INL* = 0). The two empty boxes in Figure 45 represent either the detector and exploitation module or the RAM and the computation module since the same integrity check is used for the sequential or RAM-based BIST.

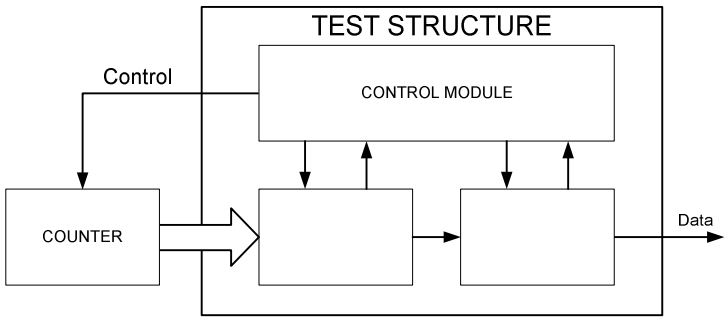


Figure 45: Integrity check of the histogram BIST structure.

In processor-based BIST, the integrity check of the test infrastructure includes the testing of the processor, timer and RAM cores, the test of the interconnects and the test of the ROM with the stored BIST program. Since most of the above activities are included in the general SoC test, only the test of the ROM (i.e., the checksum of its contents) is required in addition.

5.5 Wrapper design

The above test structures were implemented in the IEEE Std 1500 test wrapper. The IEEE Std 1500 defines a mechanisms for testing the core designs within the SoC. It foregoes addressing the analog circuits and focuses on facilitating an efficient digital test. It has serial and parallel access mechanisms and a rich set of test instructions to facilitate the interoperability between the core designers and the core integrators without compromising the intellectual properties. The IEEE Std 1500 reduces the test cost through improved automation, promotes a good design-for-test technique and improves the test quality through improved access. The test wrapper was designed in complete conformance to the rules described in standard [6]. The principal block scheme is shown in Figure 46.

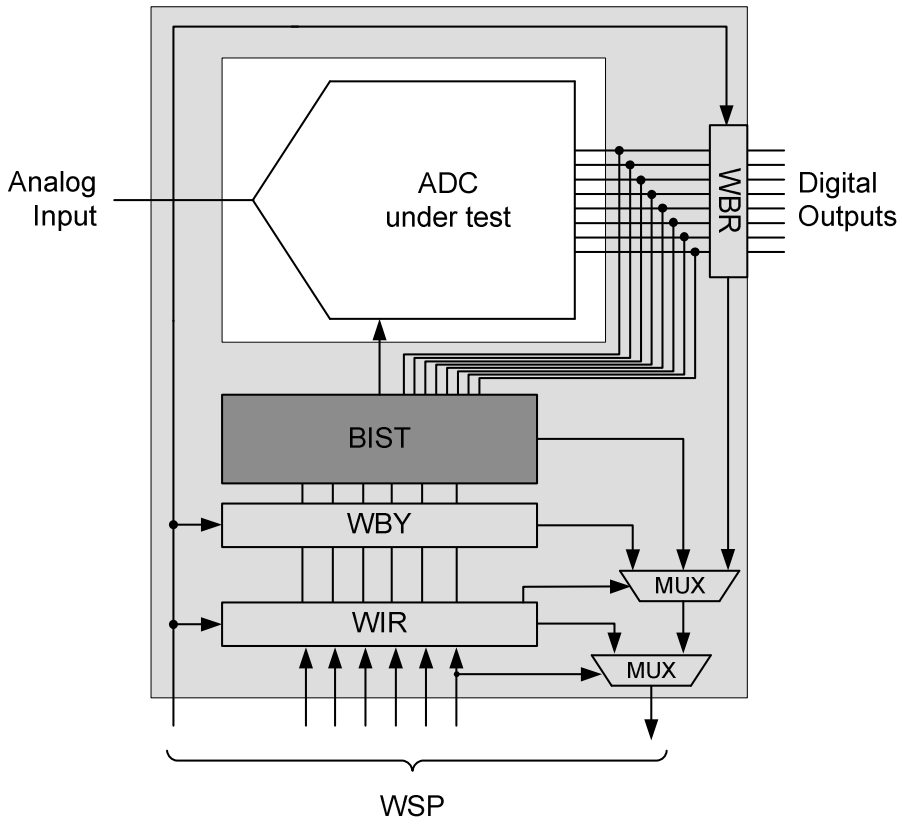


Figure 46: Histogram-based BIST structure in IEEE Std 1500 wrapper.

The shaded area represents the test wrapper logic. The histogram test structure (sequential or RAM-based) is included and denoted by the BIST block. The IEEE 1500 wrapper is composed of the wrapper serial ports (WSP), the wrapper instruction register (WIR), the wrapper bypass register (WBY) and the wrapper boundary register (WBR). These are the mandatory objects; there can also be some optional objects, like the wrapper parallel ports. The WIR instructions are serially entered into the wrapper circuitry via the WSP, which is presented in Figure 47.

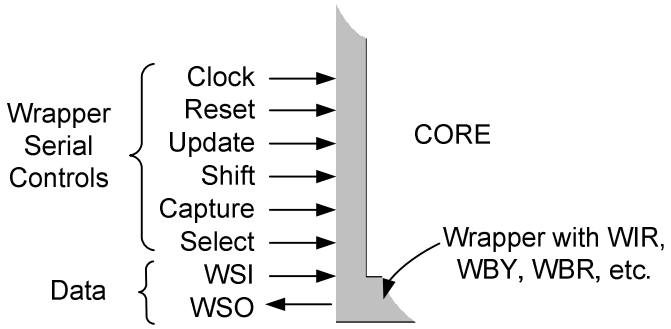


Figure 47: Wrapper serial port (WSP).

The WIR contains a shift stage, an instruction decode and an update stage, as depicted in Figure 48. The WIR has to be unconditionally accessible by the WSP. The WIR is controlled and clocked by the standard WSP terminals.

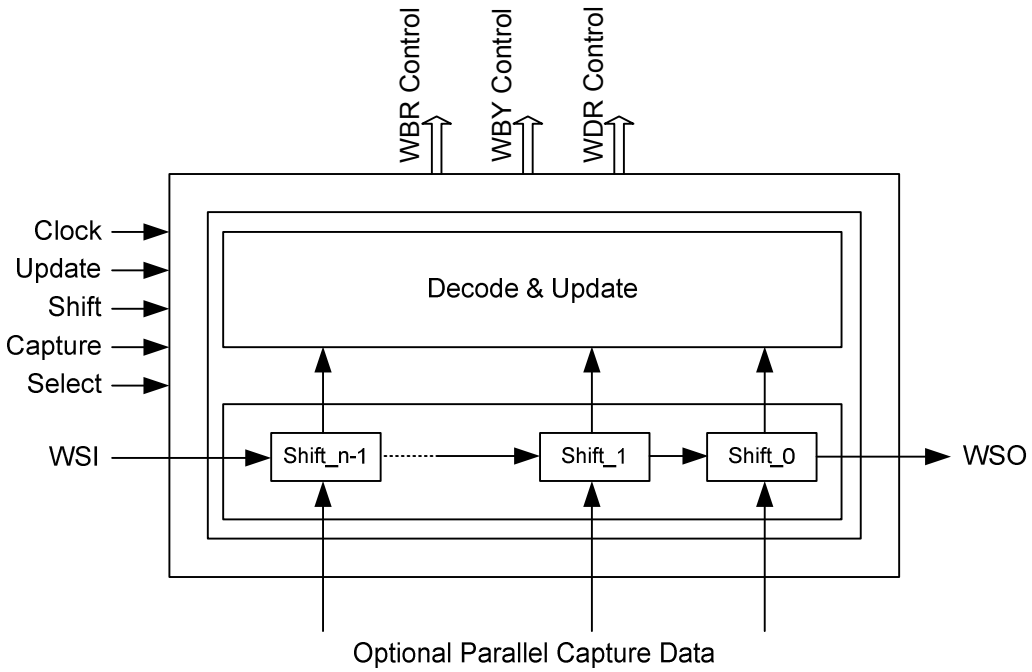


Figure 48: Wrapper instruction register (WIR).

The circuitry in the WIR must generate the necessary controls to enable and control the operational modes of the WBR. The WIR must also provide signals for selecting the WBR and the WBY between the WSI and the WSO. The WIR was designed according to the standard, so that the data shifted into the WIR shift registers does not affect the currently active wrapper and core modes, until a WIR update operation occurs.

The IEEE 1500 wrapper has to contain a WBY, as shown in Figure 49. The WBY connects the WSI and WSO whenever the WIR contains the bypass instruction. The WBY must include at least one bit of serial shift data. There must be no inversion of the logic values when shifting the data between the WSI and the WSO.

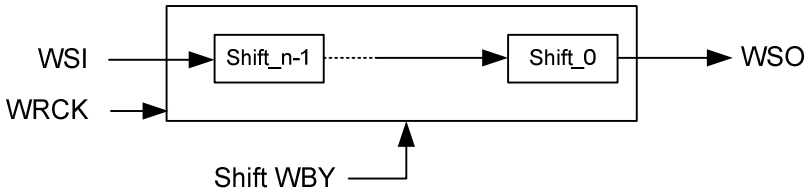


Figure 49: Wrapper bypass register (WBY).

In the case of a multi-core chip the wrappers of the individual cores are in a serial chain. The bypass function is used when a particular core does not need to be tested, but others do. The input data is then simply bypassed to the cores under test.

One of the differences between IEEE Std 1500 and IEEE Std 1149.1 is that IEEE Std 1500 supports not only serial interfaces, but also a parallel interface for accessing a wrapped core. The serial interface is characterized by a single chain composed of all the WBR cells. The parallel interface is characterized by an arbitrary number of WBR cells surrounding the core. The WBR is constructed of WBR cells that have four data terminals: the cell functional input (CFI), the cell functional output (CFO), the core test input (CTI) and the core test output (CTO). The Figure 50 presents the WBR register surrounding the core, the single WBR cell terminals and the single WBR cell structure.

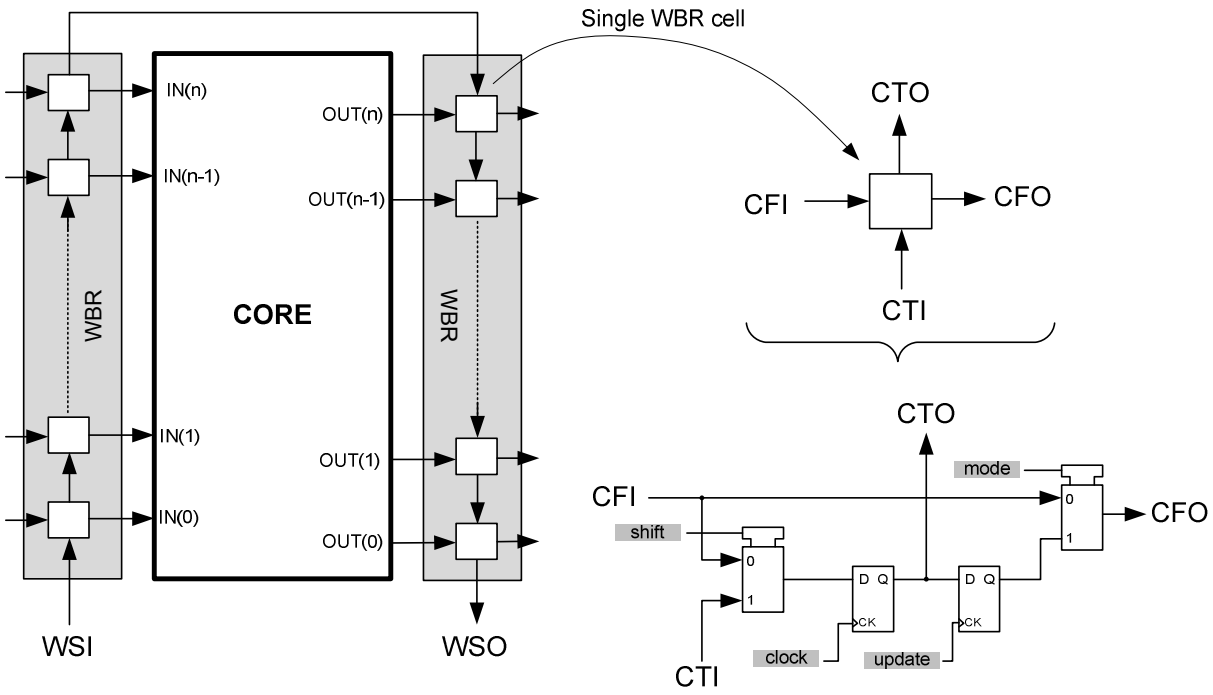


Figure 50: Wrapper boundary register (WBR).

The WBR serial shift path is formed by connecting the WBR cells CTO-to-CTI. A WBR operation is triggered by with events. An event is an uninterrupted, predefined sequence of one or more steps. The predefined events are shift, update, transfer, capture and apply.

Shift is an event whereby the data stored in the WBR shift path are advanced one storage position closer to the WSO.

Capture indicates an event where the value present on the CFI, is stored in a sequential element within the WBR cell.

Update is an optional event whereby data stored in a WBR cell's shift path are loaded in an off-shift-path storage element.

Transfer is an optional event that moves data to or within the shift-path of a WBR cell. The purposes of the transfer event are to properly position the captured data in the shift path for observation and to provide stimulus data if needed.

Apply is a derivative event which causes test data to be applied from input cells onto core inputs or to be applied from the output cell onto the WBR functional outputs, with regard to the wrapper facing mode (inward or outward facing mode), respectively.

The ADC input is not included in the wrapper since the IEEE Std 1500 is only defined for digital SoC. By analogy with IEEE Std 1149.1/IEEE Std 1149.4 we introduced an analog boundary module at the ADC input. For this purpose, our proprietary test chip [75] was employed. The analog boundary module (ABM) and its test bus interface circuit (TBIC) are added to the wrapper, as shown in the Figure 51. Since they are controlled by the 1500 wrapper control signals they could actually be regarded as an analog extension of the IEEE Std 1500 infrastructure.

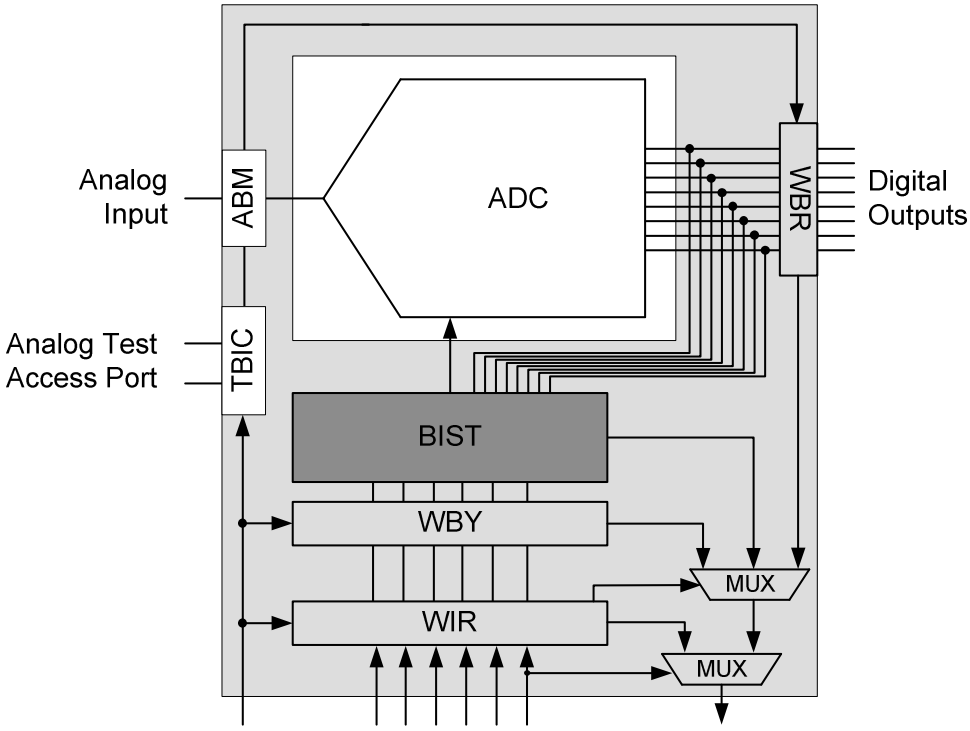


Figure 51: Extension of the IEEE Std 1500 with ABM and TBIC modules.

In test mode the input stimulus is conveyed to the ADC through the test bus interface circuit (TBIC) and the analog boundary module (ABM) of the IEEE Std 1149.4 logic. The ABM presented in Figure 52 is the heart of the standard framework for a mixed signal test.

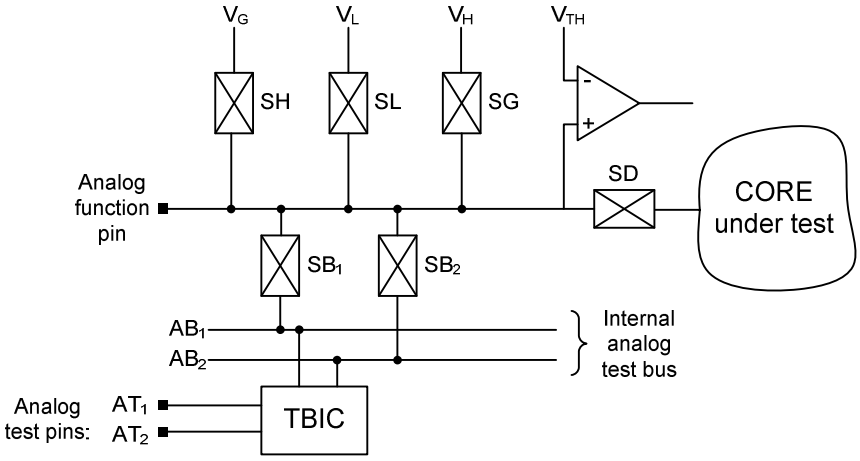


Figure 52: Analog boundary module (ABM) circuitry.

The ABM is designed to be used as “virtual probes” so that the analog test pins $AT_{1,2}$ can be connected to the analog function pin without the need for physical probing. But in our case we needed to connect the AT_1 to the core (to apply the input stimulus to the ADC core). This was achieved using a modified ABM connection (i.e., the ABM’s analog function pin and core pin are interchanged), as depicted in Figure 53.

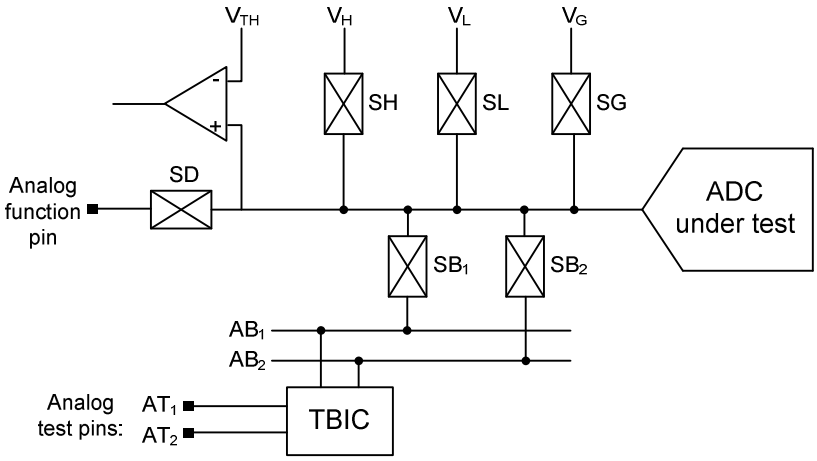


Figure 53: Inverted use of ABM.

The ABM's switch SD enabled us to disconnect the analog function pin from the ADC and so allow us to apply the stimulus only to the input of the ADC under test. To evaluate the influence of the ABM MOS switches on the measurement accuracy, two situations were measured: input stimuli applied via ABM and TBIC wrapper circuitry; input stimuli applied directly to ADC as shown in Figure 51 and 54, respectively.

In order to reduce the congestion of the test control interface in the SoC and to consolidate the operation of the IEEE Std 1500 wrapper with the IEEE Std 1149.1 based standards we also implemented a straightforward interface between the IEEE Std 1500 and the IEEE Std 1149.4. In this case the use of a test access port (TAP) controller is required. The detailed implementation of the ADC as a unit under test and the test wrappers are presented in Figure 54.

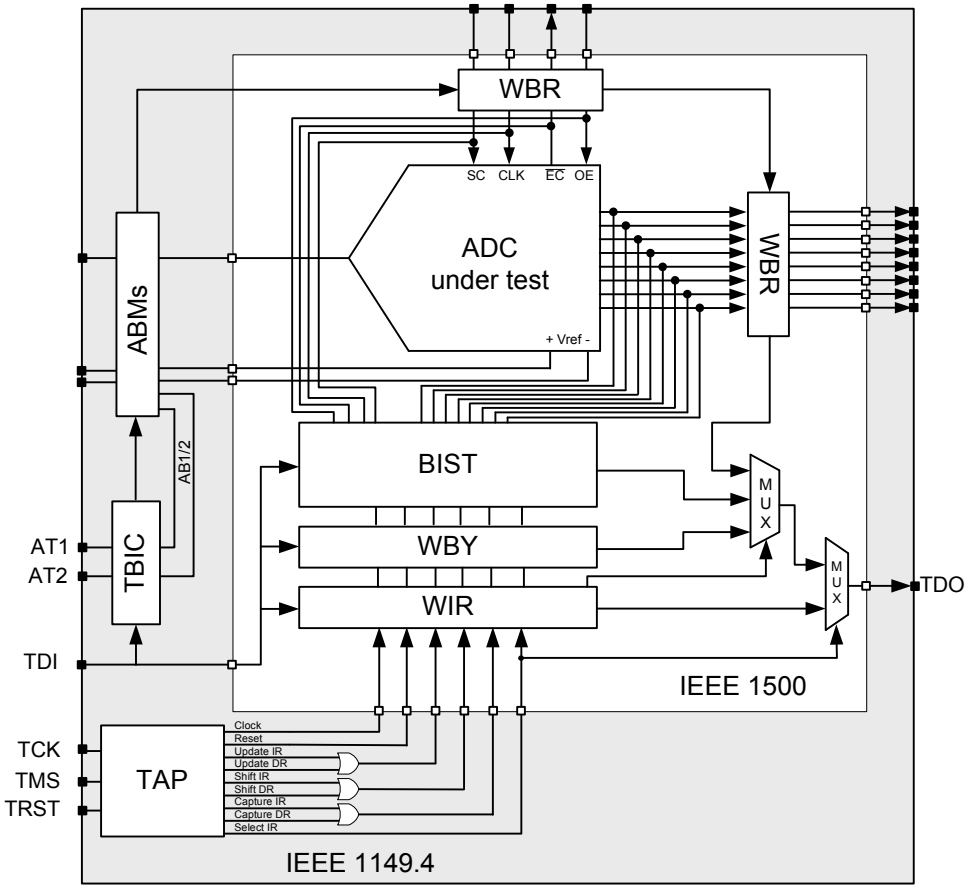


Figure 54: Implemented test wrappers with ADC as core under test.

5.6 Performance and experimental results

The IEEE Std 1500 test wrapper with the above-described BIST structures was implemented in a Spartan3 XC3S200 FPGA. An off-the-shelf ADC (an 8-bit ADC CMOS microprocessor compatible) was chosen as the unit-under-test. The measured parameters of the employed ADC conformed with the specifications provided by the manufacturer, which demonstrates the feasibility of the histogram-based test technique using the implemented test structures. Let us now compare the presented solutions from different points of view that may serve as the basis for the selection of a suitable strategy for the histogram BIST implementation of the embedded core.

The input stimulus (triangle wave) was generated by a high-resolution (16-bit) DAC. By applying the hig-resolution DAC a lower quantization error and a higher signal-to-noise ratio (SNR) were achieved. For the given measurement conditions (triangle input signal, given sampling frequency) and ideal ADC characteristics, we determined the ideal count of the non-extreme codes H_{ideal}. The measurement uncertainty Δ (expressed in terms of LSB) of the histogram base test is defined by:

$$\Delta = \frac{1}{H_{ideal}} \ . \tag{22}$$

The quantization error of an 8-bit ADC is 50dB, the measured SNR of the generated triangle input signal was more than 96dB, hence its effect on the measurement uncertainty is negligible.

To prevent sample leakage into the adjacent code bins, while constructing the histogram, a coherent sampling condition is required. In order to achieve coherent sampling, the DAC sample clock was fed into the test structure.

In the sequential BIST approach, if the test signal is generated such that the number of occurrences of the two extreme codes was equal to the number of occurrences of the non-extreme codes, the number of samples required for processing a single ADC code is

$$N = H_{ideal} \cdot 2^n \ . \tag{23}$$

From this it follows that the number of samples required for the processing offset error, gain error, DNL and INL:

$$N_{offset} = H_{ideal} \cdot 2^n = N \ , \tag{24}$$

$$N_{gain} = m \cdot H_{ideal} \cdot 2^n = m \cdot N \ , \tag{25}$$

$$N_{DNL} = (2^n - 2) \cdot H_{ideal} \cdot 2^n = (2^n - 2) \cdot N \ , \tag{26}$$

$$N_{INL} = (2^n - 2) \cdot H_{ideal} \cdot 2^n = (2^n - 2) \cdot N \ . \tag{27}$$

where n is the number of ADC bits, and m is the number of central codes on which the gain calculation is performed, respectively.

The number of all the samples needed for a complete measurement with a sequential histogram approach is

$$N_{all} = (1 + m + 2^{n+1} - 4) \cdot N \ . \tag{28}$$

In the RAM-based BIST and the processor-based BIST approaches all the codes are processed simultaneously. The number of samples required for a complete measurement is

$$N_{all} = N = H_{ideal} \cdot 2^n . \tag{29}$$

Table 2 presents the measurement uncertainty Δ and the overall test time of the approaches for different values of H_{ideal} at 5 µs sample time.

Table 2: Measurement uncertainty and overall test time of the sequential, RAM-based and processor-based BIST

	measurement	Test time [s] for 8-bit ADC		
H_{ideal}:	uncertainty Δ [LSB]:	Sequential BIST:	RAM-based BIST:	processor-based BIST:
4	0.250	3.37	0.725	0.729
8	0.125	6.08	0.732	0.838
16	0.063	11.49	0.745	0.867
32	0.031	22.33	0.772	0.990
64	0.016	43.99	0.825	1.362

Notice that the test time in the case of the sequential BIST is much higher than that of the RAM-based BIST. While the test time of the 8-bit ADC at the measurement uncertainty of 0.250 [LSB] (Least Significant Bit) may still be acceptable, the situation gets worse when testing ADC cores with resolution higher than 8-bit. Table 3 shows the test time of the sequential, RAM-based and processor-based BIST for an 8, 10, 12 and 16-bit ADC. While the test time of the sequential BIST for a 10-bit ADC may still be acceptable in some particular cases, test times of higher-bit ADCs are obviously not acceptable in practice.

Table 3: Test time of sequential and RAM-based BIST for different ADCs

number of	Test time [s] (H_{ideal}=4)		
ADC bits:	Sequential BIST:	RAM-based BIST:	processor-based BIST:
8	3.37	0.725	0.729
10	44.18	0.907	0.922
12	697.69	1.161	1.278
16	178543.32	3.116	3.222

RAM-based and processor-based BISTs are much faster than a sequential BIST, but there are other issues that should be considered. As regards the hardware overhead, the RAM-based BIST employs much more resources than the sequential BIST. We synthesized approaches with the Cadence RTL Compiler and the resulting designs are shown below. In order to get a realistic view of the hardware overhead of individual solutions, sequential and RAM-based BIST logic implementations without the IEEE Std 1500 wrapper logic are summarized in Table 4.

Table 4: Sequential and RAM-based BIST implementations (without IEEE Std 1500 wrapper logic).

number of	Sequential BIST			RAM-based BIST		
ADC bits:	logic gates:	flip-flops:	RAM cells:	logic gates:	flip-flops:	RAM cells:
8	881	233	-	461	156	2560
10	911	249	-	541	172	12288
12	961	265	-	583	188	57344
16	1067	297	-	684	220	1179648

Next, the resources of the complete BIST implementations, including the wrapper control logic of the presented solutions are summarized in Table 5.

Table 5: BIST implementations including the IEEE Std 1500 wrapper control logic.

	Sequential BIST			RAM-based BIST			Processor-based BIST		
number of ADC bits:	logic gates:	flip-flops:	RAM cells:	logic gates:	flip-flops:	RAM cells:	logic gates:	flip-flops	RAM cells:
8	1012	293	-	594	216	2560	63	59	-
10	1063	319	-	612	242	12288	71	72	-
12	1131	345	-	680	268	57344	97	95	-
16	1208	371	-	889	320	1179648	205	127	-

The processor-based BIST has a minimal hardware overhead since the majority of its tasks are performed by the cores already included in a SoC. However, such a solution is specific for the given processor core and is not easily portable to other SoCs. On the other hand, Sequential BIST and RAM-based BIST are autonomous and can be ported together with an ADC core to other designs.

A scalability test can address different aspects. The analysis of the BIST solutions for *n*-bit ADC cores shows that by increasing *n* the hardware overhead of the sequential BIST increases linearly, while it exhibits exponential growth in the case of the RAM-based BIST and practically does not change in the processor-based BIST. The test time of the sequential BIST grows exponentially.

Providing high-quality stimulus signals is crucial in order to accomplish an efficient and consistent BIST of a mixed-signal core. The signal can be internally generated in the test wrapper, or supplied via TAM, either from an external signal generator or generated by another core (if available) in the SoC-under-test. In our case, including a triangle-wave generator [76, 77] in the test wrapper would considerably increase the complexity, and hence an externally supplied stimulus is a preferred approach.

Laboratory measurements were performed with the test solutions presented. The impact of MOS switches connecting the external source to the ADC core was simulated by introducing an ABM module of an IEEE Std 1149.4 compliant test chip. In Table 6 the results for the sequential BIST approach are presented, Table 7 summarizes the results using the RAM-based BIST technique and Table 8 presents result for the processor-based BIST.

Table 6: Measured characteristics for the sequential BIST approach.

	Number	Offset error [LSB]:		Gain error [LSB]:		DNL [LSB]:		INL [LSB]:	
H_{ideal}	of samples:	via ABM:	direct:	via ABM:	direct:	via ABM:	direct:	via ABM:	direct:
4	537600	0.50	-0.32	0.70	0.00	0.50	0.50	0.94	0.25
8	1075200	0.25	-0.34	0.34	0.05	0.38	0.38	0.75	0.25
16	2150400	0.50	-0.29	0.52	0.12	0.31	0.31	0.53	0.19
32	4300800	0.78	-0.30	0.30	0.06	0.28	0.25	0.36	0.06
64	8601600	0.45	-0.28	0.22	0.12	0.25	0.23	0.67	0.14
128	17203200	0.71	-0.30	0.24	0.13	0.23	0.22	0.40	0.18
256	34406400	0.43	-0.27	0.22	0.15	0.39	0.21	0.48	0.23
512	68812800	0.22	-0.28	0.05	0.11	0.24	0.11	0.48	0.20

Table 7: Measured characteristics for the RAM-based approach.

H_{ideal}	Number of samples:	Offset error [LSB]:		Gain error [LSB]:		DNL [LSB]:		INL [LSB]:	
		via ABM:	direct:	via ABM:	direct:	via ABM:	direct:	via ABM:	direct:
4	1024	0.75	-0.38	0.50	0.05	0.50	0.50	0.58	0.25
8	2048	0.43	-0.44	0.28	0.03	0.38	0.38	0.29	0.23
16	4096	0.40	-0.41	0.34	0.10	0.27	0.19	0.36	0.15
32	8192	0.39	-0.36	0.26	0.13	0.22	0.20	0.57	0.21
64	16384	0.35	-0.38	0.32	0.08	0.23	0.21	0.34	0.15
128	32768	0.40	-0.39	0.29	0.14	0.24	0.15	0.44	0.17
256	65536	0.35	-0.36	0.29	0.11	0.22	0.20	0.52	0.20
512	131072	0.36	-0.35	0.24	0.10	0.23	0.18	0.33	0.19

Table 8: Measured characteristics for the processor-based approach.

H_{ideal}	Number of samples:	Offset error [LSB]:		Gain error [LSB]:		DNL [LSB]:		INL [LSB]:	
		via ABM:	direct:	via ABM:	direct:	via ABM:	direct:	via ABM:	direct:
4	1024	0.50	-0.34	0.50	0.00	0.50	0.50	0.25	0.50
8	2048	0.35	-0.36	0.50	0.00	0.32	0.38	0.56	0.45
16	4096	0.32	-0.41	0.52	0.06	0.29	0.31	0.34	0.48
32	8192	0.44	-0.29	0.39	0.05	0.26	0.25	0.42	0.46
64	16384	0.37	-0.36	0.32	0.06	0.24	0.23	0.62	0.34
128	32768	0.26	-0.37	0.41	0.06	0.40	0.22	0.41	0.36
256	65536	0.28	-0.29	0.36	0.05	0.25	0.21	0.47	0.43
512	131072	0.31	-0.37	0.34	0.04	0.28	0.11	0.45	0.39

The measured parameters were in good agreement with the values obtained by the conventional technique using laboratory measurement equipment. The measured and calculated static performance characteristics of the employed ADC also conform with the specifications provided by the manufacturer, which gives confidence to the described approaches.

6 Oscillation-based ADC BIST

The oscillation-based test (OBT) [38, 45, 46] is a low-cost analog and mixed-signal test technique primarily used for fault detection. In the test mode, the circuit is transformed into an oscillator and the frequency of the oscillation is measured and compared to a reference value obtained on a known-good circuit operating in the same conditions. Assuming that most possible faults manifest themselves in the discrepancy from the reference oscillation frequency, the technique offers an effective means of fault detection. It requires minimal reconfiguration of the circuit-under-test and is thus suitable for a built-in self-test. The oscillation-based test has been applied to different kinds of circuits including filters, ADC and DAC, PLLs, etc. [10, 44, 49-51, 54, 78-80]. An extensive summary of the proposed solutions can be found in [8, 43].

In the development phase of the test procedure for the target unit-under-test, the impact of possible faults on the measured frequency must be analyzed. If some faulty components do not affect the frequency of oscillation, additional measurements of other parameters are required in order to achieve the required fault coverage. The acceptance/reject margins are determined on the basis of the required user specifications. For example, in an industrial application of a go/no-go test of a low-pass filter stage as a part of a communication unit was produced by Hipot Hyb [81], the test system was first adjusted in reference to a golden filter stage. Reference filter stages with values corresponding to the acceptance ranges within ±0.1% and ±0.3% were measured in the test system in order to set up the test-system acceptance margins. A pre-production series of 546 active adjusted filter stages were tested, and 95% passed the go/no-go test. The stages that passed the go/no-go test were assembled in the final product. A functional test confirmed that all the circuits operated within the required tolerances.

Simulations can be performed to assess the impact of the discrepancy from the nominal values of individual components of the unit-under-test on the oscillation frequency. Some work has been done to improve the sensitivity of the OBT to parametric faults [54]. A so-called predictive oscillation based test proposed in [82, 83] can be used to define the acceptance region for an OBT implementation.

So far, most of the OBT related work has been directed either towards the problem of modification of the given circuit-under-test into an oscillator, or towards an analysis of the detected faults (exploring ways to increase the fault coverage or even trying to perform a fault diagnosis). Little attention has been paid to the measurement accuracy of the developed OBT solutions. As stated in [84], for a simple arrangement consisting of a zero-crossing detector and a counter, the measurement accuracy is determined by the oversampling ratio ($T_{oscillation}/T_{sampling}$) and depends of the phase shift, which is equivalent to one sampling period ($T_{sampling}$) as a maximum. While this is in general true for any OBT implementation, there exist other sources of inaccuracy associated with the OBT implementations adapted for the ADC under-test [12].

6.1 ADC OBT environment

The basic OBT arrangement for testing an analog-to-digital converter (ADC) is shown in Figure 55. It consists of a feedback loop, which forces the ADC to oscillate around a selected code [10]. The input stimulus of the ADC is a triangle-wave signal of symmetrical slope controlled by the OBT control logic. The measured frequency of the triangle-wave signal can be used for a simple go/no-go test or for determining static ADC parameters such as the differential nonlinearity (DNL) and the integral nonlinearity (INL).

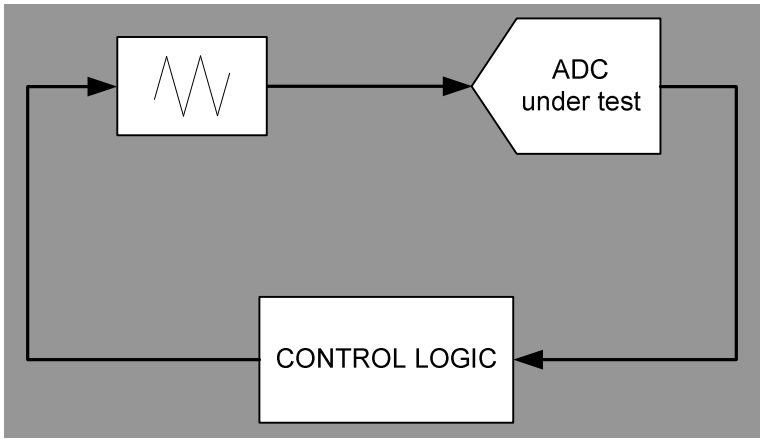


Figure 55: Basic oscillation-base test set up configuration.

An ADC generates a single output code for a range of input voltages. The voltage level where the ADC output changes from code i-1 to code i is denoted as the threshold voltage (voltage transition) $V_{TH}(i)$, and where the ADC output changes from code i to code i+1 as the threshold voltage $V_{TH}(i+1)$, respectively. The difference between the threshold voltages $V_{TH}(i)$ and $V_{TH}(i+1)$ is denoted as the code width $Q(i)$. The code widths of an ideal n-bit ADC are equal. Their width is obtained by dividing the full-scale range value (FSR) by the number of ADC codes, as defined in equation (8).

$$Q_N(i) = Q_N \tag{30}$$

In the case of an ideal ADC, an instantaneous ADC conversion and instantaneous operation of the OBT structure the oscillation signal would oscillate only in the range of the selected code with the oscillation period T_{osc}, as presented in Figure 56.

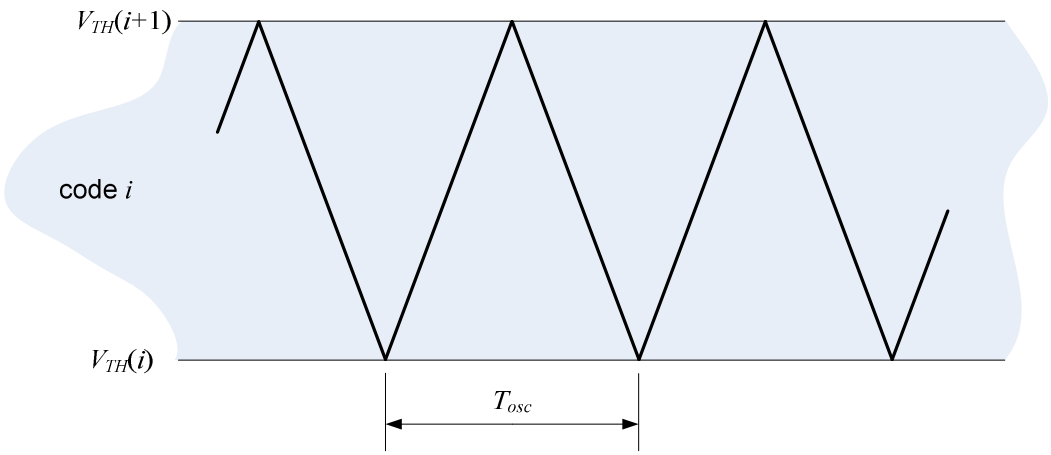


Figure 56: Oscillating in ideal circumstances.

In case of the described circumstances the code width $Q(i)$ could be calculated from the oscillation period (frequency) as:

$$Q(i) = S \cdot \frac{1}{2} \cdot T_{osc} . \tag{31}$$

where S is the slope of the input signal.

In real life, the conversion of an ADC is not instantaneous; the acquired code is delayed for the conversion time. This is also presumed in the case of an ideal ADC. The conversion time of the ADC presents the lower boundary for sampling time. In order to ensure the synchronized operation the output of ADC is usually updated at the next sampling time (the ADC output is delayed for one sample time). In the OBT the output of the ADC is compared to the selected code i at the sampling instances. The first sample acquired outside the threshold voltages of the selected code i results in an ADC output change at the next sample and thus the oscillation signal slope inversion, as depicted in 56.

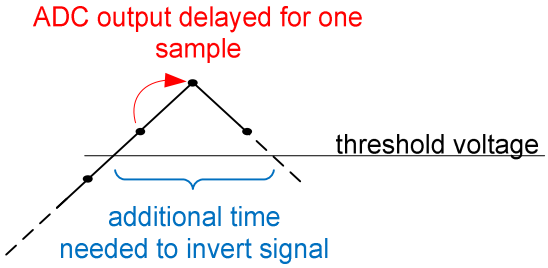


Figure 57: Signal inversion in OBT.

In other words, the slope of the triangle-wave signal is inverted one sample time after the sampled input signal attains or crosses the threshold voltage. The ADC delay results in the additional time, which must be considered when calculating the code width from the measured oscillation frequency with the OBT.

The DNL measures the relative deviation of each code from the ideal value (1 LSB). The deviation is due to the fact that the actual threshold voltages differ from their nominal values. For a given code i DNL is defined with equation (11). The slope of the input signal is defined with the signal generator (integrator) and can be arbitrary. If the slope S of the oscillating triangle-wave signal is such that the signal traverses the code width Q in a k number of samples time T_S, we can define:

$$Q(i) = S \cdot k \cdot T_S . \tag{32}$$

The actual code width $Q(i)$ is obtained by determining the time $k \cdot T_S$ of the input signal passing between $V_{TH}(i)$ and $V_{TH}(i+1)$. The evaluation of k from the measured oscillation frequency represents the basis of the OBT technique for ADC.

The oscillation period is a multiple of the sampling period, because the slope alterations are performed only at the sampling instances. Consequently, the oscillation period can be exactly determined. This comes with a price, notably an interval of slopes produce the same oscillation frequency, which leads to the measurement uncertainty.

In the case where the slope is such that the input signal traverses the code width in a multiple of sampling periods, the phase shift of the input signal is the same at each threshold voltage. Consequently, the oscillation period is constant, irrespective of the phase shift. In the case where the input signal traverses the code width in a non-whole number of sample periods ($k \notin \mathbb{N}$), the phase shift θ'_{i+1} crossing the voltage threshold $V_{TH}(i+1)$ differs from the phase shift θ_i crossing the voltage threshold $V_{TH}(i)$ (depicted by the dashed line in Figure 58).

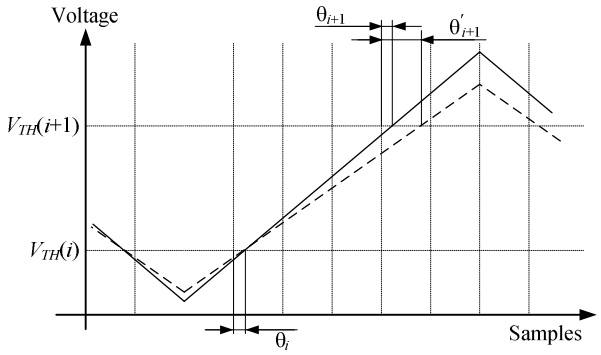


Figure 58: Phase shift of the oscillating ADC input signal.

By investigating the phenomenon more closely, we noticed that by varying the phase shift θ_i two distinct oscillation periods occur. To determine the DNL value from the oscillation frequency let us examine the ADC sampling of the oscillating signal in detail.

The oscillated triangle-wave signal is sampled periodically, as shown in Figure 59. Let us denote the first sample time, where the input signal is equal to or greater than the threshold voltage $V_{TH}(i+1)$ as $T(-1)$. Since the output code of the ADC is delayed for the sample time T_S, the slope of the input signal is inverted at the sample time $T(0)$.

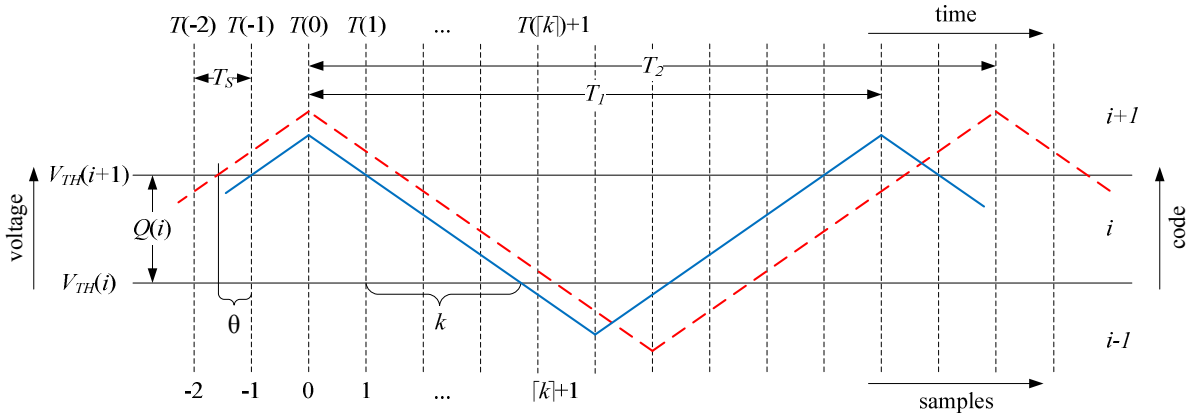


Figure 59: Detailed timing diagram of the ADC sampling in OBT.

In general, the transition of the input signal over the threshold voltage occurs at $T(-1)$ or somewhere within the interval $T(-2)$ and $T(-1)$. For a further analysis let us denote the time between the transition of the threshold voltage $V_{TH}(i+1)$ and the sample time $T(-1)$ as the time shift $\theta \cdot T_S$, where the values of θ are:

$$0 \le \theta < 1. \tag{33}$$

The time from the transition of the threshold voltage $V_{TH}(i+1)$ to the inversion of the slope at $T(0)$ is $(1+\theta)\cdot T_S$. Since the rising and the falling slopes are equal, the transition of the threshold voltage $V_{TH}(i)$ occurs at $(1+\theta+k)\cdot T_S$ after $T(0)$. Depending on the values of k and θ, we distinguish two different cases:

$$k + \theta < \lceil k \rceil \tag{34}$$

and

$$k + \theta \ge \lceil k \rceil, \tag{35}$$

where ⌈ ⌉ is the ceiling operator which rounds a number to its nearest integer value in an upwards direction. This results in two different signals, depicted Figure 59, with the continuous-blue and the dashed-red line. The first case refers to the situation where the signal transition of the threshold voltage $V_{TH}(i)$ occurs before $T(\lceil k \rceil+1)$, while the second case refers to the situation where the signal transition of the threshold voltage V_{TH} (i) occurs after $T(\lceil k \rceil+1)$. Consequently, the inversion of the slope of the two signals differs for one sample time, which leads to two different oscillation periods:

$$T_1 = (2 \cdot \lceil k \rceil + 4) \cdot T_s, \tag{36}$$

$$T_2 = (2 \cdot \lceil k \rceil + 6) \cdot T_s. \tag{37}$$

The actual measured oscillation period T is either T_1 or T_2. Because we cannot distinguish between the two cases, the computation of k from the expressions (36) and (37) gives

$$k = \frac{T}{2 \cdot T_s} - 3 \pm 1. \tag{38}$$

The two possible outcomes reflect in the code width calculation as:

$$Q(i) = S \cdot \left(\frac{T}{2 \cdot T_S} - 3 \right) \cdot T_s \pm S \cdot T_S. \tag{39}$$

In the case of an ideal ADC we can assume, without the loss of generality, that the slope S of the triangle-wave signal is generated such that the signal traverses the nominal code width Q_N in a whole number k_N of sample periods T_S.

$$Q_N = S \cdot k_N \cdot T_S \tag{40}$$

$$k_N \in \mathbb{N} \tag{41}$$

From that it follows only one nominal oscillation period, denoted by T_N:

$$T_N = (2 \cdot k_N + 6) \cdot T_s. \tag{42}$$

Since we cannot distinguish between the two cases described in (36) and (37) the ambiguity of oscillation frequency is manifested in the DNL estimation as:

$$DNL(i) = \frac{T - T_N}{2 \cdot k \cdot T_s} \pm \frac{1}{k} \tag{43}$$

The derived equations show an inherent measurement uncertainty Δ which has to be considered when determining the ADC nonlinearity from the oscillating frequency.

$$\Delta = \pm \frac{1}{k} \tag{44}$$

The measurement uncertainty of the classical OBT techniques is related to the uncertainty of the measurement of the oscillation period and is determined by the oversampling ratio $T_{oscillating}/T_{sampling}$. In the case of ADC OBT, however, the measurement of the oscillation period is precise, but due to the arbitrary input signal phase shift and the non-coherent input signal two oscillation periods are possible. This in turn manifests itself in the measurement uncertainty, as demonstrated.

6.2 Performance and experimental results

In order to analyze the influence of different measurement parameters on the DNL estimation using the OBT method a simulation environment using Matlab Simulink was developed. The environment (shown in Figure 60) allows us to perform calculations of DNL over the whole set of ADC codes with an arbitrary number of oscillation periods at an arbitrary code. Since the ADC block in the Matlab Simulink environment models an ideal ADC, an additional block with a nonlinear transfer function was added in front of the ADC to mimic the realistic non-linear behavior. Furthermore, a noise generator was included, which allows us to introduce noise into the input signal.

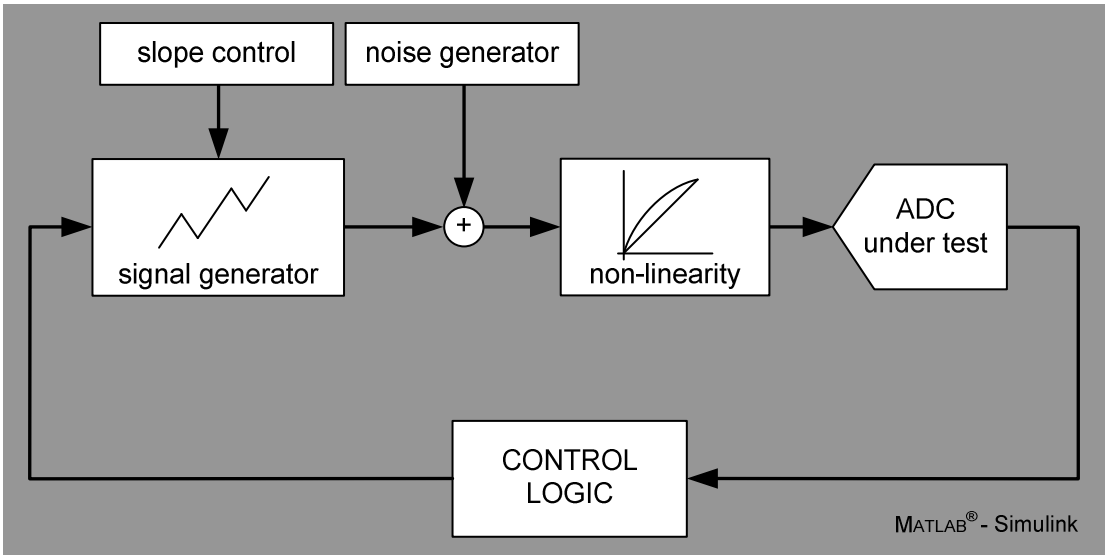


Figure 60: The environment for analyzing the OBT of ADC.

The slope of the input signal is the most significant measurement parameter. A higher slope increases the oscillation frequency and shortens the measurement time; however, it affects the measurement accuracy. To analyze the behavior of the testing method with different input signal slopes, slope control was added to the input signal generator.

In the ideal case where the slope of the input signal is time invariant and without any noise, the oscillation period at a given code is constant (because the durations of the input signal's rise and fall are constant). Hence, the measurement of half of the period is sufficient for a determination of the oscillation frequency. However, in the presence of noise the oscillation periods may vary. The influence of noise on the oscillation frequency can be reduced by measuring and averaging over several periods. This, on the other hand, prolongs the DNL measurement time. The feedback-loop control logic was adopted to perform an arbitrary number of oscillations at a given code, which enables us to study all the possible scenarios.

In order to measure the DNL of the ADC, the DNL of each code, with the exception of the extreme codes, must be determined. To achieve this, while keeping the measurement time as short as possible, the feedback control logic is designed in such a way that the DNLs of successive codes are determined in their order, starting with the code 1. The subsequence of the generated input signal with two oscillations per code is shown in Figure 61.

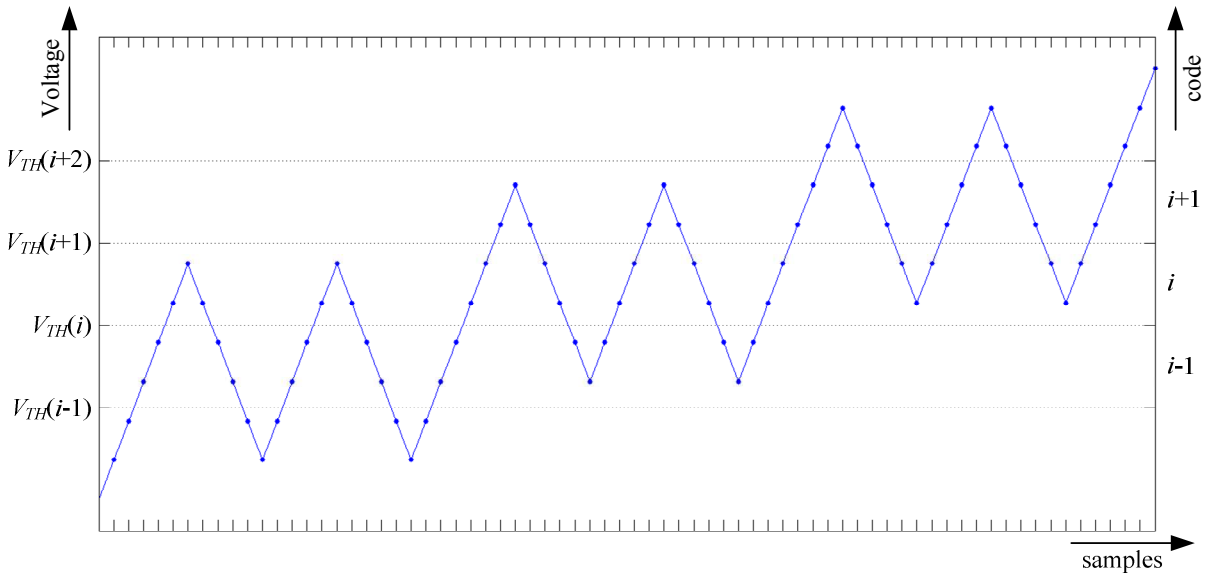


Figure 61: The signal in the OBT procedure.

Using the developed Matlab Simulink environment we performed numerous experiments measuring the DNL of an ideal 8-bit ADC with different input signal slopes. The DNL is measured for all non-extreme codes. In Figure 62 the DNL measurement in the case of coherent sampling (with k=2) is depicted. The upper bound of the measurement uncertainty is depicted by a dashed line. As expected, the determined DNL of each code is exact and is equal to 0.

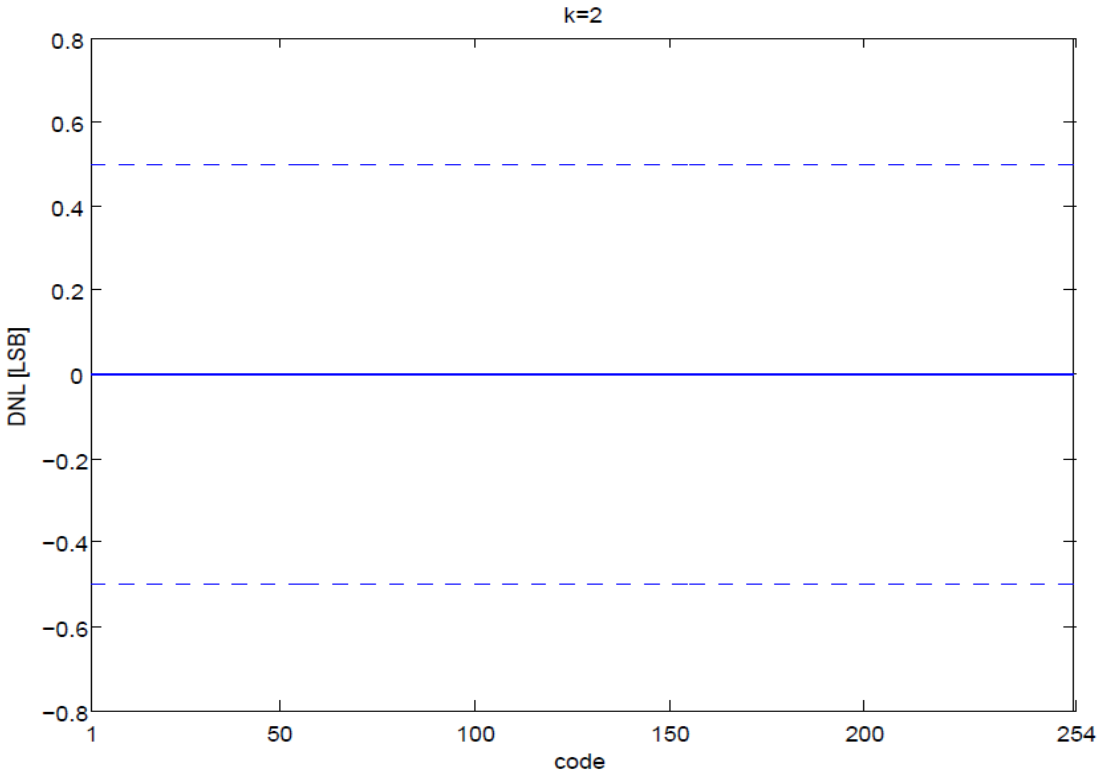


Figure 62: The DNL measurement of an ideal ADC using an input slope such that k=2.

Figures 63, 64, and 65 present DNL measurement using input slopes with k equal to 2.1, 2.5, and 2.9, respectively. While the slope of the input signal remains the same during the experiment and the code widths are equal in the case of an ideal ADC, the phase shifts for each code differ. This manifests itself in two different oscillation periods in the whole ADC

range and consequently in two different DNL values. While the DNL for the ideal ADC is 0, the measured DNL deviates from the exact value. The dashed line represents the upper bound of the measurement uncertainty.

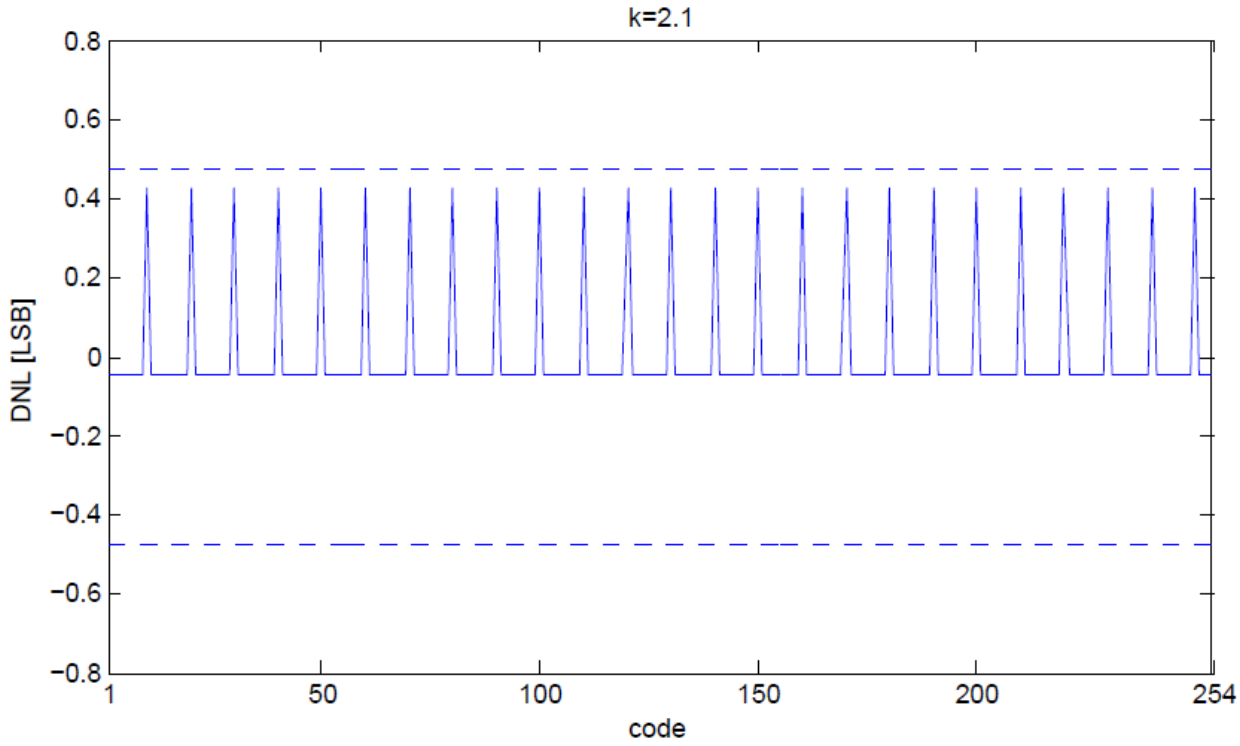


Figure 63: The DNL measurement of an ideal ADC using an input slope such that k=2.1.

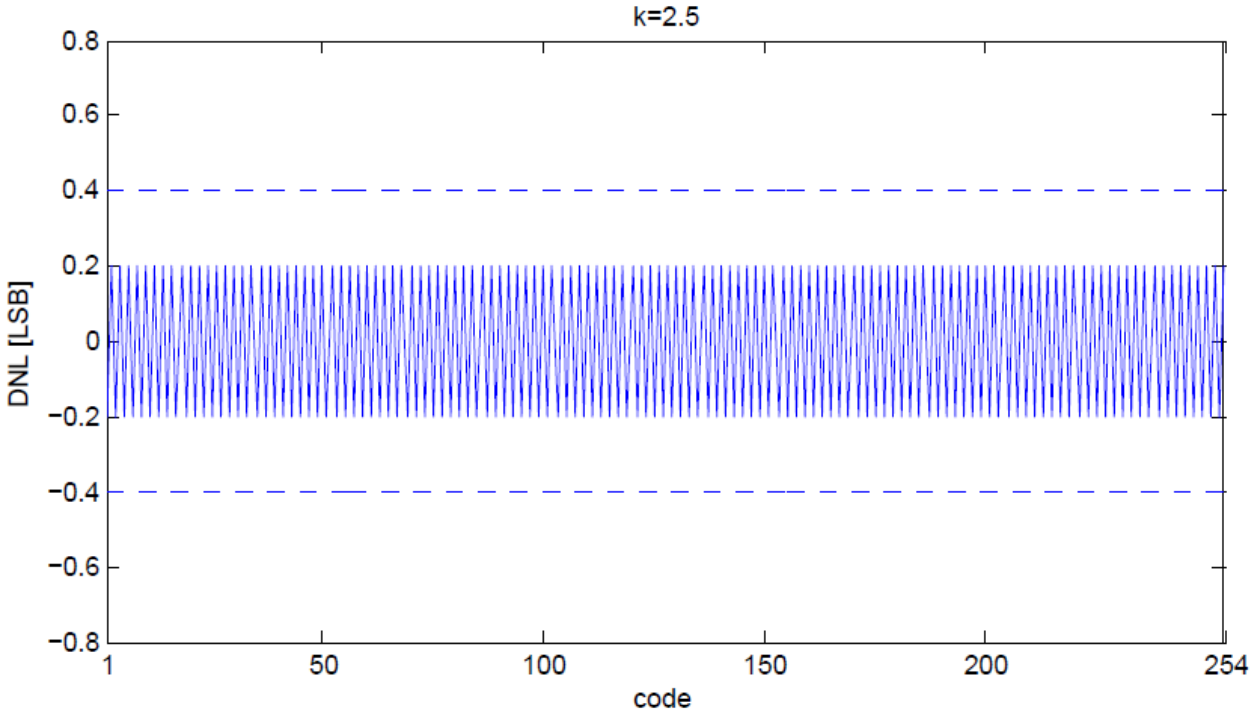


Figure 64: The DNL measurement of an ideal ADC using an input slope such that k=2.5.

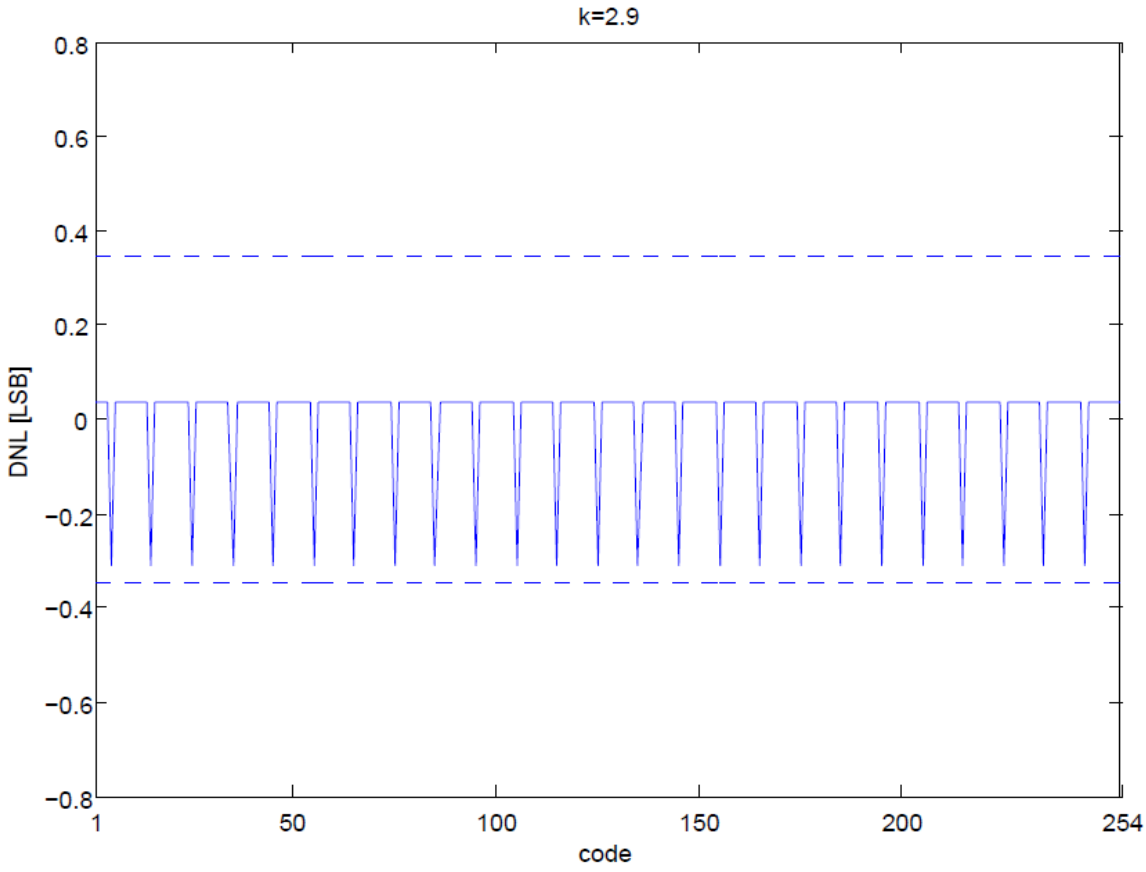


Figure 65: The DNL measurement of an ideal ADC using an input slope such that k=2.9.

The dependency between the input signal slope expressed by k (the number of samples required for traversing the code width) and the measured DNL of an ideal ADC is depicted in Figure 66. The DNL of the ADC is composed of the maximum values of the DNLs of the individual codes. The upper bound of the DNL measurement uncertainty is depicted by the solid line. The measured DNLs for the considered slopes are indicated by marks.

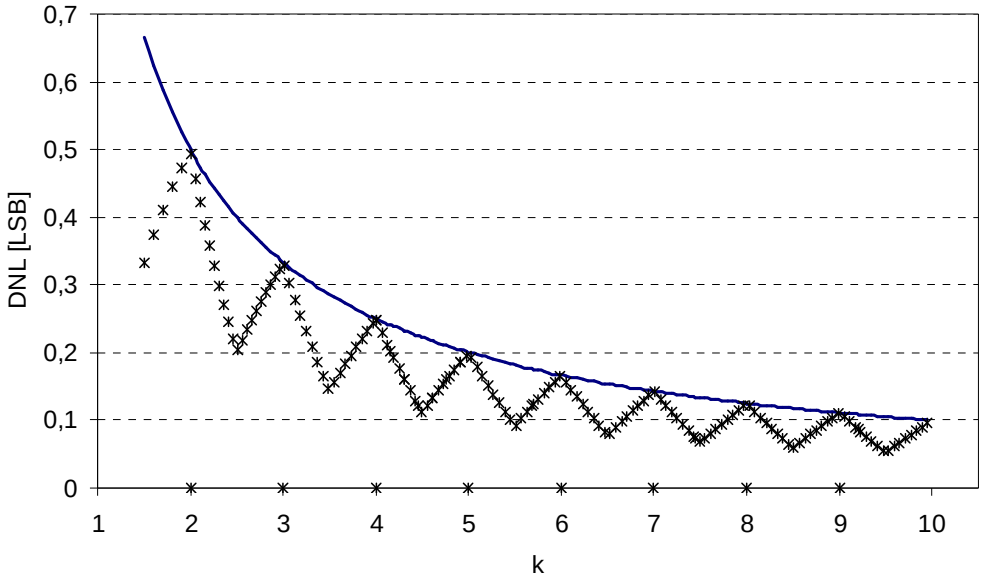


Figure 66: Measured DNL of an ideal ADC using OBT with different input slopes.

For an illustration, an 8-bit ADC with *DNL* = 0.3 LSB was simulated. The OBT technique was modeled as describe above in the Matlab Simulink environment. The oscillation-based test for different values of *k* and θ was performed. The obtained values of *DNL* and the corresponding measurement uncertainty are given in Table 8.

Table 9: Measured DNL values with the OBT technique.

θ / *k*	0.062	0.188	0.312	0.438	0.562	0.688	0.812	0.938	measurement uncertainty:
	measured *DNL* [LSB]								
2	0.00	0.00	0.00	0.50	0.50	0.50	0.50	0.50	0.500
3	0.00	0.33	0.33	0.33	0.33	0.33	0.33	0.33	0.333
4	0.25	0.25	0.25	0.25	0.25	0.25	0.50	0.50	0.250
6	0.17	0.17	0.33	0.33	0.33	0.33	0.33	0.33	0.167
8	0.25	0.25	0.25	0.25	0.25	0.38	0.38	0.38	0.125
16	0.25	0.25	0.31	0.31	0.31	0.31	0.31	0.31	0.062
32	0.28	0.28	0.28	0.31	0.31	0.31	0.31	0.31	0.031

The oscillation-based test of the ADC manifests itself in two different oscillation periods depending on the time of the transition of the threshold voltage relative to the sampling.

7 Test-sequence optimization in the SoC test and diagnosis

SoC design integrates large, reusable blocks (i.e. cores) that have been designed and verified in earlier applications in practice. Embedded cores provide a wide range of functions, like CPUs, DSPs, ADCs, DACs, interfaces, controllers, memories, and others. The cores put together in a SoC normally originate from different core providers. In order to protect their intellectual property, core providers do not completely reveal design and implementation details, which makes the problem of SoC testing rather challenging to the core user (i.e., SoC designer). Adherence to IEEE Std 1500 may help both the core user and the core provider to establish common means for developing efficient test solutions. However, in production test or system maintenance, fault detection may not be sufficient. Faults must be isolated to determine the actual cause and the origin of the defects in order to modify the production process and/or repair the product. In system maintenance, detected failures must be diagnosed and repaired as rapidly as possible to return the system to the correct operation. Fault isolation usually requires more thorough and costly procedures than fault detection. It is therefore imperative to find low-cost solutions to the diagnosis problem.

System operation in the presence of faults can be presented by different system-failure states. The goal of the diagnostic procedure is to identify the actual failure state by executing a sequence of tests which provide some information on system behavior. In principle, any measurement, signal, or other observable event can be viewed as a test.

Consider a test with a binary outcome (pass or fail) in such a diagnostic procedure. Assume that before the test is executed, the system is in a system state $s_i \epsilon S$, where S denotes the set of current candidate system states. The test splits the set S into subsets A and B. Let A denote the resulting set of candidate system states when the test fails and B is the set of candidate system states when the test passes. For a symmetrical test: $A \cap B = 0$. In other words, the test fails if and only if the system is in the faulty state $s_i \epsilon A$. Hence, if a symmetrical test passes, system states A are omitted from the set of the candidate faulty system states. The symmetrical test thus has a 100% fault coverage for the set A of candidate faulty system states. For an asymmetrical test: $A \cap B \neq 0$. An example of an asymmetrical test is the alarm indicator on a control panel. If the light is on, the actual system is one of the candidate faulty system states of A. On the other hand, if the light is not on, one cannot distinguish between the two cases: either there is no alarm or the indicator bulb is bad.

Determining the sequence of tests required to reach a diagnostic conclusion at minimum cost is known as the test-sequencing problem [85]. A tool that generates solutions to the generalized test sequencing problem has been implemented at Jožef Stefan Institute and a collection of examples covering different test-variant situations is provided. The examples offer a quick way to get acquainted with the tool; however, its application in practice is somewhat more complicated.

The initial problem is to provide an appropriate set of tests that would enable the desired diagnostic resolution. In SoC testing, a set of tests for testing individual cores are specified by the designer. For the cores conforming to IEEE Std 1500, the tests can be initiated and the test stimuli/results transmitted via TAM. We can also assume that the tests of interconnections among the cores conforming to IEEE Std 1500 are available. For the remaining embedded cores and glue logic we refer to the work of Murray [86] and the notion of test propagation through system modules. During a hierarchical system design, a collection of tests is generated for fault detection in individual system parts. The transparency analysis of these

pre-computed tests is performed in the next step in order to organize them in an efficient overall system test. While Murray [86] actually addresses the problem of fault detection in a complex system, we extend the notion of transparency for the purpose of a system diagnosis. While the original pre-computed tests may be sufficient for fault detection, serial chains of functional blocks make the problem of fault localization rather difficult. Modifications (i.e., extensions) of the original sets of pre-computed tests are normally needed to achieve adequate controllability and observability for system diagnosis. In practice, different approaches can be taken, introducing, for example, additional lines for accessing internal points from primary inputs or outputs, or even adding scan chain logic to the original design.

In order to assess the applicability of the diagnostic tool in SoC fault localization we performed selected case studies on electronic sub-systems that might imitate, from the complexity point of view, some glue logic in a SoC. We found that even for relatively small examples, such as the computational element of radix-4 pipeline Cooley-Tukey FFT module [13], the description of individual tests and the determination of their associated costs and probabilities of detecting faults requires a substantial effort. The *a priori* probability data of system states and the probability data of asymmetric test outcomes are needed for the generation of test sequences. They can be assessed or computed from MIL failure-rate data tables. Another way to obtain this data is by collecting the diagnoses through the system life cycle. In this way, initially estimated probabilities are improving on the reliability during system operation.

$$p_b = \left(\frac{n}{n+m}\right) \cdot \frac{r}{n} + \left(\frac{m}{n+m}\right) \cdot p_a \,. \tag{45}$$

The *m*-estimation approach can be used to combine an initial estimation and the accumulated knowledge about system failures. The parameter m in the above equation is a weight, by which one can regulate the impact of the initially estimated probabilities p_a on the updated probabilities p_b, and the impact of the relative frequency r/n, with n trials and r successes.

Lessons learned from the performed case studies indicate that in our case the generation of optimal diagnostic test sequences for a SoC with complex embedded cores would require substantial effort in the preparation of the required input test data for the diagnosis tool. So we are rather focusing on the trade-off between hardware overhead and test time in a BIST of a mixed-signal core implemented in a IEEE Std 1500 wrapper. Test strategies based on the experimental results of different implementation approaches of the histogram technique are described in [11], recently revised for publication with minor corrections required to the journal Computing and Informatics.

8 Conclusions

In this thesis we address the problem of developing BIST structures for embedded ADC cores in a SoC. In a conventional test approach, the unit-under-test is connected to external automatic test equipment (ATE), which generates test vectors and collects test responses. However, for a SoC with complex embedded cores a huge amount of test data needs to be transferred between the ATE and the core-under-test, which drastically increases the test time. A viable alternative is to implement a built-in self-test (BIST) in the target SoC. The embedded BIST logic takes the role of the ATE by generating test vectors and analyzing test responses. In this way, the communication bottleneck between the ATE and the SoC is avoided, only the remaining low-speed operations required for the execution of the complete test of the core-under-test (such as initiation of test mode, start of BIST and evaluation of test results) are left to the ATE which can be managed by a low-cost tester. IEEE Standard 1500 provides an efficient test infrastructure for testing digital cores; however, its applications in mixed-signal core test remain an open issue. So far, no thorough analyses supported by empirical results from wrapper implementations for mixed-signal cores in practice have been reported.

In this thesis, we developed three extreme BIST solutions for testing an embedded ADC core using the popular histogram-based technique. The implemented test logic complies with the IEEE Std 1500. The first BIST configuration employs test wrapper logic with the minimum hardware overhead. The second solution aims at a test wrapper optimized for the minimum test time. In the third solution, the processor core performs the ADC core test and computes the required test parameters. The three approaches can be generalized to other types of mixed-signal cores, as follows:

- the test wrapper performs the core test and evaluates the results on the fly,
- the results (i.e., measurements) are stored in RAM and processed in batch,
- the other cores of the SoC (if available) take the role of testing.

In a purely digital SoC, the first two approaches are preferred since in most cases the stimuli are generated by a LFSR (linear-feedback shift register) logic and the test results are collected by a MISR (multiple input signature register) based on the same principles. This logic is compact and scalable and provides portability of design.

The BIST of a SoC with mixed-signal cores faces different problems. Test methods possibly implemented in a test wrapper require specific logic, which differs considerably from one case to another. There are no common parts; everything must be conceived from scratch. In fact, since a mixed-signal core is put in a SoC with a well-defined target application, it is prudent to consider the possible reuse of the application software (i.e., device drivers) in a processor-based BIST.

A processor-based BIST is traditionally employed in testing embedded memories. Its application in a mixed-signal core test has not been widely investigated in the literature, although proprietary solutions in practice no doubt exist. In this regard, the comparison of the empirical data presented in the thesis allows an assessment of the pros and cons of the alternative approaches and may serve as a guideline in similar applications.

We also studied the OBT approach for a possible application in a BIST solution of the embedded ADC core in a SoC. In order to analyze the influence of different measurement

parameters on the DNL estimation using the OBT method a simulation environment using Matlab Simulink was developed. We investigated the measurement conditions and noticed an inherent measurement uncertainty, which has to be considered when deriving the parameters from the oscillation frequency. We further elaborated this issue and derived a theoretical background for computing the measurement uncertainty of the approach.

Finally, we performed some selected case studies in order to assess the applicability of a diagnostic tool developed at the Jožef Stefan Institute in SoC fault localization. The lessons learned from the performed case studies indicate that the generation of the optimal diagnostic test sequences for a SoC with complex embedded cores requires substantial effort in the preparation of the required input test data for the diagnosis tool. The problem, however, is interesting and remains the subject of ongoing research.

The main contributions of the thesis can be summarized as follows:

- We developed three histogram-based BIST solutions for testing an embedded ADC core in a SoC, pursuing the goals of a minimized hardware overhead (sequential BIST), minimized test time (RAM based BIST) and scalability (processor-based BIST).
- So far, no thorough analysis supported by empirical results from IEEE Std 1500 wrapper implementations of a mixed-signal test in practice has been reported. In this regard, the above BIST solutions conforming to the IEEE Std 1500 represent a notable contribution and a guideline for the implementation of other mixed-signal test techniques in a IEEE Std 1500 test wrapper.
- We have shown that the oscillation-based test of the ADC manifests itself in two different oscillation periods depending on the time of the transition of the threshold voltage relative to the sampling. Since it is not possible to distinguish between the two cases, the ambiguity of the frequency of oscillation should be considered in the OBT implementations. We demonstrated that the measurement uncertainty of the DNL due to the above ambiguity is 1/k.

9 Acknowledgements

It is an honor for me to thank the people who made this thesis possible.

I would like to express my utmost gratitude toward my supervisor, Prof. Dr. Franc Novak, who besides his experienced guidance in the scientific world has offered me his support in a number of ways.

Special thanks goes to Assist. Prof. Dr. Anton Biasizzo, without his expertise and help, it would be next to impossible to write our research papers.

I am grateful to my colleagues at the Computer Systems Department for their enthusiasm and inspiration during my research work at the Jožef Stefan Institute.

I would like thank Gorenje, d.d. who enabled me to study and my co-workers at the Gorenje company for the intriguing discussions.

I am also indebted to my family and my dearest Katja, for their never-ending support.

10 References

[1] Burns, M. and Roberts, G. R. *An introduction to a mixed-signal IC test and measurement* (Oxford University Press, 2001).

[2] Agrawal, V. D. Design of mixed-signal systems for testability. *Integration, the VLSI journal* 26, pp. 141-150 (1998).

[3] Azaïs, F.; Bernard, S.; Bertrand, Y.; Comte, M.; Renovell, M. A-to-D converters static error detection from dynamic parameter measurement. *Microelectronics Journal* 34**,** pp. 945-953, (2003).

[4] *IEEE Standard Test Access Port and Boundary-Scan Architecture*, IEEE Std 1149.1-2001, 2001.

[5] *IEEE Standard for a Mixed-Signal Test Bus*, IEEE Std 1149.4-1999, 1999.

[6] *IEEE Standard Testability Method for Embedded Core-based Integrated Circuits*, IEEE Std 1500-2005, 2005.

[7] Stroud, C. E. *A Designer's Guide to Built-In Self-Test*, (Kluwer Academic Publishers, Dordrecht, 2002).

[8] Huertas, J. L. (ed.) *Test and design-for-testability in mixed-signal integrated circuits* (Kluwer Academic Publishers, Dordrecht, 2004).

[9] Corcoran, J. J.; Hornak, T., Skov P. B. A high resolution error plotter for analog-to-digital converters, *IEEE Transactions on Instrumentation and Measurement*, 24, 4, pp. 370-374, 1975.

[10] Arabi, K.; Kaminska, B. Oscillation Built-In Self Test (OBIST) Scheme for Functional and Structural Testing of Analog and Mixed-Signal Integrated Circuits, *Proceedings of the IEEE International Test Conference*, pp. 786-795, 1997.

[11] Novak, F.; Mrak, P.; Biasizzo A. Test strategies for embedded ADC cores in a system-on-chip, a case study, *Computing and Informatics*, accepted for publication, (2011).

[12] Mrak, P.; Biasizzo, A.; Novak, F. Measurement accuracy of oscillation-based test of analog-to-digital converters, *ETRI Journal*, vol. 32, no. 1, pp. 154-156 (2010)

[13] Mrak, P.; Novak, F.; Biasizzo, A. Generation of sequential diagnostic procedures including symmetrical, asymmetrical and multi-valued tests, *International Review on Computers and Software*, vol. 4, no.1, pp. 113-118, (2009)

[14] Davis., B. *The Economics of Automatic Testing*, (McGRAW-HILL Book Company (UK) Limited, London, 1982).

[15] Moore, G. E. Cramming more components onto integrated circuits, *Electronics*, vol. 38, no. 8, 1965.

[16] Dubash, M. Moore's Law is dead, says Gordon Moore, http://news.techworld.com/operating-systems/3477/moores-law-is-dead-says-gordon-moore/, Techworld, 2005, (accessed January 2010).

[17] Intel Corporation Excerpts from A Conversation with Gordon Moore: Moore's Law, ftp://download.intel.com/museum/Moores_Law/Video-Transcripts/Excepts_A_Conversation_with_Gordon_Moore.pdf, Copyright Intel Corporation, 2005, (accessed January 20010).

[18] Landman, B. S., Russo, R. L. On a Pin Versus Block Relationship For Partitions of Logic Graphs, *IEEE Transactions on Computers*, vol. c-20, no. 12, pp. 1469-1479, (1971).

[19] International Technology Roadmap for Semiconductors, Test and Test Equipment, 2009 edition.

[20] Taylor, D.; Evans, P. S. A.; Pritchard T. I. Testing of mixed-signal systems using dynamic stimuli, *Electronic Letters* vol.29, no. 9, (1993).

[21] Shannon, C. E. Communication in the Presence of Noise, *Proceedings of the Institute of Radio Engineers*, vol. 37, no.1, pp. 10-21, (1949). Reprinted in: *Proceedings of the IEEE*, vol. 86, no. 2, (1998).

[22] Nyquist, H. Certain topics in telegraph transmission theory, *Transactions of the A.I.E.E.*, vol. 47, pp. 617-644, (1928). Reprinted in: *Proceedings of the IEEE*, vol. 90, no. 2, (2002).

[23] Jerri, A. J. The Shannon sampling theorem – its various extensions and applications: A tutorial review. *Proceedings of the IEEE*, vol. 65, no. 11, (1977).

[24] Mahoney, M. *DSP-based testing of analog and mixed-signal circuits* (Wiley-IEEE Computer Society Press, Washington 1987).

[25] Sinclair, J. (ed.) *Collins Cobuild English language dictionary* (Collins Publishers, London, 1992)

[26] Gizopoulos, D. (ed.) *Advances in Electronic Testing: Challenges and Methodologies* (Springer, Dordrecht, 2006)

[27] Cooper, W. D. *Electronic Instrumentation and Measurement Techniques* (2nd edition, Prentice Hall, New Jersey, 1978).

[28] Graf, R. F. *Modern Dictionary of Electronics*, (7th edition, Newnes, Boston, 1999).

[29] Baker, M. *Demystifying mixed-signal test methods*, (Newnes, Boston, 2003).

[30] Wang, L.; Stroud, C. E.; Touba, N. A. *System-on-Chip Test Architectures: Nanometer Design for Testability* (Morgan Kaufmann Publishers, Burlington, 2008).

[31] Grout I. A. *Integrated circuit test engineering: modern techniques* (Springer, London, 2006).

[32] *ILAC-G8:1996 Guidelines on Assessment and Reporting of Compliance with Specification*, International Laboratory Accreditation Cooperation, 1996.

[33] *ISO14253-1 Geometrical Product Specifications (GPS) -- Inspection by measurement of workpieces and measuring equipment -- Part 1: Decision rules for proving conformance or non-conformance with specifications*, International Organization for Standarditization, 1998

[34] *M3003 The Expression of Uncertainty and Confidence in Measurement*, United Kingdom Accreditation Service, 2007

[35] Gray, N. ABCs of ADCs Analog-to-digital converter basics. www.national.com/appinfo/adc/files/ABCs_of_ADCs.pdf 2006 (accessed: february 2010).

[36] Maloberti, F. *Data converters* (Springer, Dordrecht, 2008).

[37] Schreier, R.; Temes, G. C. *Understanding Delta-Sigma Data Converters*, (Wiley, New Jersey, 2005).

[38] Arabi, K.; Kaminska, B., Oscillation-Test Strategy for Analog and Mixed-Signal Circuits, *Proceedings VLSI Test Symposium*, pp. 476-482, (1996).

[39] Corcoran, J.J.; Hornak, T.; Skov, P.B. A high resolution error plotter for analog-to-digital converters, *IEEE Transactions on Instrumentation and Measurement*, pp. 763-771, (1975).

[40] S. Max, Optimum measurement of ADC code transition using a feedback loop, *Proceedings of the 16th IEEE Instrumentation and Measurement Technology Conference*, pp. 1415–1420, (1999).

[41] Zhao, Z.; Ivanov, A. Embedded servo loop for ADC linearity testing, *Microelectronics Journal*, 33, pp. 773–780, (2002).

[42] *IEEE Standard for Terminology and Test Methods for Analog-to-Digital Converter*, IEEE Std 1241-2000, 2000.

[43] Huertas Sánchez, G.; Vázquez García de la Vega, D.; Rueda Rueda, A.; Huertas Díaz, J. L. *Oscillation-Based Test in Mixed-Signal Circuits*, (Springer, Dordrecht, 2006).

[44] Huertas, G.; Vázquez, D.; Rueda, A.; Huertas, J. L. Effective Oscillation-Based Test for Application to a DTMF Filter Bank. *Proceedings of the International Test Conference*, pp. 549-555, (1999).

[45] Arabi, K.; Kaminska, B. Testing Analog and Mixed-Signal Integrated Circuits Using Oscillation-Test Method. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 16, no. 7, pp. 745-753, (1997).

[46] Arabi, K.; Kaminska, B.; Sunter, S. Design for Testability of Integrated Operational Amplifiers Using Oscillation-Test Strategy. *Proceedings of the International Conference on Computer Design*, pp. 40-45, (1996).

[47] Ihs, H.; Arabi, K.; Kaminska, B. Testing digital to Analog Converters Based on Oscillation-Test Strategy Using Sigma-delta Modulation. *IEEE International Conference on Computer Design*, (1998).

[48] Karim, A.; Kaminska, B. Parametric and Catastrophic Fault coverage of Analog Circuits in Oscillation-Test Methodology. *Proceedings of the 15th IEEE VLSI Test Symposiu*, pp. 166-171, (1997).

[49] Karim, A.; Kaminska, B.; Rzeszut, J. BIST for D/A and A/D Converters. *IEEE Design & Test of Computers* (1996).

[50] Santo Zarnik, M.; Novak, F.; Maček, S. Design of Oscillation-Based Test Structures for Active RC Filters. *IEE Proceedings – Circuits, Devices and Systems*, vol. 147, no. 5, pp. 297-302, (2000).

[51] Kač, U.; Novak, F. Oscillation Test Scheme of SC Biquad Filters Based on Internal Reconfiguration. *Journal of Electronic Testing*, vol. 23, no. 6, pp. 485-495, (2007).

[52] Santo Zarnik, M.; Novak, F.; Maček, S. Design of Oscillation-Based Test Structures for Active RC Filters. *Proccedings of European Design and Test Conference* (1997).

[53] Huertas, G.; Vázquez, D.; Rueda, A.; Huertas, J. L. Practical oscillation-based test of integrated filters. *IEEE Design & Test*, vol. 9, no. 6, (2002).

[54] Huertas, G.; Vázquez, D.; Rueda, A.; Huertas, J. L. Testing Mixed-Signal Cores: A Practical Oscillation-Based Test in an Analog Macrocell. *IEEE Design & Test*, vol. 9, no. 6, (2002).

[55] Frish, A.; Almy T. HABIST: Histogram-based analog built in self-test, *International Test Conference*, pp. 760-767, (1997).

[56] Azais, F.; Bernard, S.; Bertrand, Y.; Renovell, M. Implementation of linear histogram BIST for ADCs. *Proceedings of Design Automation and Test in Europe*, pp. 590-595, (2001).

[57] Renovell, M.;Azais, F.; Bernard, S.; Bertrand, Y. Hardware resource minimization for a histogram-based ADC BIST, *Proceedings of the 18th IEEE VLSI Test Symposium*, 2000.

[58] Doernberg, J.; Lee, H.S.; Hodges, D.A. Full-speed testing of A/D converters. *IEEE Journal of Solid State Circuits* 19, pp. 820-827, (1984).

[59] Vanden Bossche, M.; Schoukens, J.; Renneboog, J.; Dynamic testing and diagnostics of A/D converters. *IEEE Transactions on Circuits and Systems*, 33, pp. 775-785, (1986).

[60] Wagdy, M.F.; Awad, S.S. Determining ADC effective number of bits via histogram testing. *IEEE Transactions on Instrumentation and Measurement*, 40, pp. 770-772, (1991).

[61] Carbone, P.; Chiorboli, G. ADC sinewave histogram testing with quasi-coherent sampling. *IEEE Transactions on Instrumentation and Measurement*, 50, pp. 949-953, (2001).

[62] Blair, J. Histogram measurement of ADC nonlinearities using sine waves. *IEEE Transactions on Instrumentation and Measurement*, 43, pp.373-383, (1994).

[63] Alegria, F.C.; Serra, A.C. Error in the estimation of transition voltages with the standard histogram test of ADCs. *Measurement*, 35, pp. 389-397, (2004).

[64] F Alegria, F.C.; Serra, A.C. Overdrive in the standard histogram test of ADCs. *Measurement*, 35, pp. 381-387, (2004).

[65] F Alegria, F.C.; Serra, A.C. Standard Histogram Test Precision of ADC Gain and Offset Error Estimation. *IEEE Transactions on Instrumentation and Measurement*, 56, pp. 1527-1531 (2007).

[66] Hsin-Wen, T.; Bin-Da, L.; Soon-Jyh, H.; Histogram Based Testing Strategy for ADC. *15th Asian Test Symposium*, pp. 51-54 (2006).

[67] Alegria, F.C.; Arpaia, P.; Daponte, P.; Serra, A.C. An ADC histogram test based on small-amplitude waves. *Measurement*, 31, pp. 271-279, (2002).

[68] Serra, A.C.; Alegria, F.; Martins, R.; da Silva, M.F. Analog-to-digital converter testing--new proposals. *Computer Standards & Interfaces*, 26, pp. 3-13, (2004).

[69] Renovell, M.; Azais, F.; Bernard, S.; Bertrand, S. Hardware resource minimization for a histogram-based ADC BIST. *18th IEEE VLSI Test Symposium Proceedings*, pp. 247-252 (2000).

[70] Azais, F.; Bernard, S.; Betrand, Y.; Renovell, M. Towards an ADC BIST scheme using the histogram test technique. *IEEE European Test Workshop Proceedings*, pp. 53-58, (2000).

[71] Azais, F.; Bernard, S.; Betrand, Y.; Renovell, M. Implementation of linear histogram BIST for ADCs, Proceedings DATE'01 (2001) 590-595.

[72] Roberts, G.W.; Lu, A.K. *Analog Signal Generation for Built-in Self-test of Mixed Signal Integrated Circuits*, (Kluwer Academic Publishers, Norwell, 1995).

[73] Yong-sheng, W.; Jin-xiang, W.; Feng-chang, L.; Yi-zheng, Y. Optimal Schemes for ADC BIST Based on Histogram. *14th Asian Test Symposium*, pp. 52-57, (2005).

[74] Braga, J.A.; Machado da Silva, J.; Alves, J.C.; Matos, J.S. A Wrapper for Testing Analogue to Digital Converters Cores in SoCs. *9th IEEE European Test Symposium*, pp. 170-175, (2004).

[75] Kac, U.; Novak, F.; Azais, F.; Nouet, P.; Renovell, M. Extending IEEE Std 1149.4 Analog Boundary Modules to Enhance Mixed-Signal Test. *IEEE Design & Test of Computers*, 20, pp. 32-39, (2003).

[76] Azais, F.; Bernard, S.; Betrand, Y.; Renovell, M. Analog BIST generator for ADC testing. *Microelectronics Journal*, vol. 33, no. 10, pp. 781-789, (2002).

[77] Negreiros, M.; Carro, L.; Susin, A. A. Testing analog circuits using spectral analysis. *Microelectronics Journal*, vol. 34, no.10, pp. 937-944, (2003).

[78] Huertas, G.; Vázquez, D.; Rueda, A.; Huertas, J. L. Oscillation-based test in bandpass oversampled A/D converters. *Microelectronics Journal*, vol. 34, no.10, pp. 927-936, (2003).

[79] Ehsanian, M.; Kaminska B.; Arabi, K. A new on-chip digital BIST for analog-to-digital converters. *Microelectronics and Reliability*, vol. 38, no. 3, pp. 409-420, (1998).

[80] Romero, E.; Perett, G.; Huertas, G.; Vázquez, D. Test of switched-capacitor ladder filters using OBT. *Microelectronics Journal*, vol. 36, no. 12, pp. 1073-1079, (2005).

[81] Santo Zarnik, M.; Novak, F.; Maček, S. Efficient go no-go test of Active RC filters. *International. Journal of Circuit Theory and Applications*, vol. 26, pp. 523-529, (1998).

[82] Raghunathan, A.; Shin, H. J.; Abraham J. A.; Chatterjee, A. Prediction of analog performance parameters using oscillation based test. *22nd IEEE VLSI Test Symposium*, pp. 377- 382 (2004).

[83] Raghunathan, A.; Chun, J. H. P.; Abraham J. A.; Chatterjee, A. Quasi-oscillation based test for improved prediction of analog performance parameters. *International Test Conference*, pp.252-261 (2004).

[84] Vázquez, D.; Huertas, G.; Leger, G.; Rueda A.; Huertas, J. L. Practical solutions for the application of the oscillation-based-test: start-up and on-chip evaluation, *20th IEEE VLSI Test Symposium*, pp. 433-438, (2002).

[85] Pattipati, K.R.; Alexandridis, M.G. Application of heuristic search and information theory to sequential fault diagnosis. *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 20, No. 4, pp. 872-887, (1990).

[86] Murray, B. T.; Hayes, J. P. Test propagation through modules and circuits, *Proceedings of International Test Conference*, pp. 748-757, (1991).

Index of Figures

Index of Tables

Index of Algorithms

Appendix

Novak, F.; Mrak, P.; Biasizzo A. Test strategies for embedded ADC cores in a system-on-chip, a case study, Computing and Informatics, accepted for publication, (2011).

Mrak, P.; Biasizzo, A.; Novak, F. Measurement accuracy of oscillation-based test of analog-to-digital converters, ETRI Journal, vol. 32, no. 1, pp. 154-156 (2010)

Mrak, P.; Novak, F.; Biasizzo, A. Generation of sequential diagnostic procedures including symmetrical, asymmetrical and multi-valued tests, International Review on Computers and Software, vol. 4, no.1, pp. 113-118, (2009)

TEST STRATEGIES FOR EMBEDDED ADC CORES IN A SYSTEM-ON-CHIP, A CASE STUDY

Franc Novak, Peter Mrak, Anton Biasizzo

Computer Systems Department
Jožef Stefan Institute
Jamova cesta 39
1000 Ljubljana, Slovenia
e-mail: franc.novak@ijs.si, peter.mrak@ijs.si, anton.biasizzo@ijs.si

Abstract. Testing of a deeply embedded mixed-signal core in a System-on-Chip (SoC) is a challenging issue due to the communication bottleneck in accessing the core from external automatic test equipment. Consequently, in many cases the preferred approach is built-in self-test (BIST), where the major part of test activity is performed within the unit-under-test and only final results are communicated to the external tester. IEEE Standard 1500 provides efficient test infrastructure for testing digital cores however, its applications in mixed-signal core test remain an open issue. In this paper we address the problem of implementing BIST of a mixed-signal core in a IEEE Std 1500 test wrapper and discuss advantages and drawbacks of different test strategies. While the case study is focused on histogram based test of ADC, test strategies of other types of mixed-signal cores related to trade-off between performance (i.e., test time) and required resources are likely to follow similar conclusions.

Keywords: System-on-chip, built-in self-test, test strategies, mixed-signal testing, histogram test

1 INTRODUCTION

System-on-chip (SoC) design integrates large reusable blocks (i.e. cores) that have been verified in earlier applications in practice. Embedded cores provide a wide range of functions, like CPUs, DSPs, interfaces, controllers, memories, and others.

The advantage of embedding reusable cores in the design of a new product is a shortened design cycle resulting in reduced time-to-market and reduced cost.

The cores put together in a SoC normally originate from different core providers. In order to protect their intellectual property core providers do not completely reveal design and implementation details which makes the problem of SoC testing rather challenging to the core user (i.e., SoC designer). On the other hand, correct operation of a core in the target SoC is of interest of both core user and core provider. In order to provide an independent openly defined design-for-testability method to facilitate testing of integrated circuits containing embedded cores, IEEE 1500 Standard for Embedded Core Test [15] has been proposed and is now widely accepted in practice.

The main entity of the test architecture defined by IEEE Std 1500 is a test wrapper placed around each core of a SoC. Test wrapper provides an interface between the embedded core and its environment. For testing a core, a test source generating test vectors and a test sink collecting the test responses must be provided. Test access mechanism (TAM) transports test vectors from the source to the core and test responses from the core to the test sink. It also allows testing of interconnects between SoC cores. Test access architecture is sketched in Figure 1.

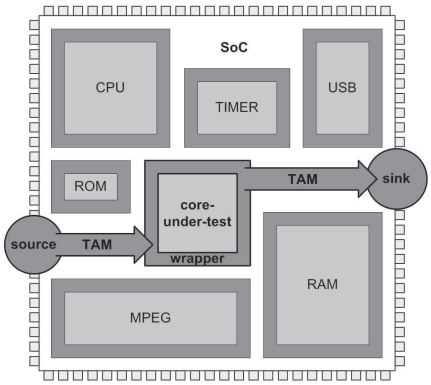


Fig. 1. Test access architecture

In a conventional test approach, TAM is connected to external automatic test equipment (ATE) which generates test vectors and collects test responses. However, for a SoC with complex embedded cores a huge amount of test data needs to be transferred between ATE and the core-under-test, which drastically increases test time. A viable alternative is to implement built-in self-test (BIST) in the target SoC. Embedded BIST logic takes the role of ATE by generating test vectors and analyzing test responses. In this way, the communication bottleneck between ATE and SoC is avoided, only the remaining low-speed operations required for the execution of the complete test of the core-under-test (such as initiation of test mode, start of BIST and evaluation of test results) are left to the ATE which can be managed by

a low-cost tester. In a typical BIST implementation, test source and test sink are implemented within the test wrapper of the tested core. In an alternative solution, some other core in the SoC may take the role of test source and/or test sink.

While the problem of optimizing test infrastructure of digital cores has been investigated by many authors [18, 22, 23, 24], its counterpart in the area of mixed-signal cores remains an open issue. So far, no thorough analysis supported by empirical results from wrapper implementations in practice has been reported. Minimization of communication with external test instrumentation, hardware overhead and test time are common yet in many cases exclusive objectives. Since a general optimization problem is almost certainly intractable it is more reasonable to expect partial solutions suitable for specific test techniques. A design-for-test architecture that allows automated test development in compliance with IEEE Std 1500 supported by in-house development tools has recently been reported by Zivkovic et al., [28]. According to the authors, the proposed approach has been successfully used for test development and characterization of mixed-signal cores in several industrial products. The described environment presents an efficient platform for comparison and evaluation of different test strategies, yet their optimization issues have not been publicly elaborated.

In this paper, an attempt is made to share experiences regarding the trade-off between hardware overhead and test time in a BIST of a mixed-signal core implemented in a IEEE Std 1500 wrapper. A case study of three extreme BIST solutions for testing an embedded analog-to-digital (ADC) core using the popular histogram-based technique [11, 14, 16] is presented. The first BIST configuration employs test wrapper logic with minimum hardware overhead. The second solution aims at test wrapper optimized for minimum test time. In the third solution, processor core performs ADC core test and computes the required test parameters. Comparison of the employed resources and the corresponding test times may serve as an aid for selection of appropriate BIST strategy.

For the preliminaries, we briefly summarize the main points of histogram based test technique with reference to previous work.

2 HISTOGRAM BASED TEST TECHNIQUE – IN BRIEF

An ADC is a device that converts an input analog voltage to a digital number proportional to the magnitude of the input signal. The resolution of the converter is equal to the number of discrete values that the converter can produce over the range of analog values. ADC performance can be verified in terms of static performance parameters such as offset, gain, differential nonlinearity (DNL) and integral nonlinearity (INL). In histogram test, a known periodic input stimulus is applied and the histogram of code occurrences (i.e., counts of individual codes) is computed over an integer number of input wave periods. From the collected results static ADC parameters can be determined. The principle is sketched in Figure 2. Histogram test has attracted research attention for more than two decades. Papers

[6, 7, 8, 9, 4, 5, 12, 13, 10, 25, 26] primarily concentrate on how the code density can be interpreted to compute the differential and integral nonlinearities, gain error and offset error, and estimate the achieved accuracy under different measurement conditions. Papers [1, 2, 20, 27] focus on implementation of histogram based test of ADCs in a BIST arrangement. A triangle-wave input signal is the most convenient for BIST of static parameter or linearity, because it can be characterized only by two values (count of extreme codes and count of other codes). The count of extreme codes is normally greater than the count of non-extreme codes due to the overflow of the input signal.

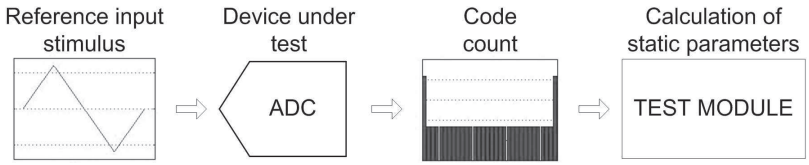


Fig. 2. The principle of histogram test with triangle-wave input signal

Coherent sampling must be provided: the ratio between the number of samples N and the number of input signal periods M must be equal to the ratio between the sampling frequency and the frequency of the input signal. In addition, the slope of the input signal is determined by the reference voltage V_{ref}, the number of ADC bits n and the number of input signal periods M.

The calculation of static parameters in the case of triangle-wave input stimulus is described in more details below in order to get a better understanding of measurements and computations performed by the BIST solutions presented in the paper.

2.1 Offset Error

The first output code transition should occur at an analog voltage that is 0.5 LSB (least significant bit) above zero. The offset error is the deviation of the first actual code transition from 0.5 LSB. In histogram based test, the offset error is calculated by the following expression

$$\text{Offset} = \frac{H(2^n - 1) - H(0)}{2H_{\text{ideal}}}, \tag{1}$$

where for a n-bit ADC: $H(2^n - 1)$ is the number of occurrences of the MSB code and $H(0)$ of the LSB code, respectively. In all expressions, H_{ideal} represents the count of each code theoretically determined for the ideal ADC and the given stimulus waveform. (Thus, H_{ideal} is an analytically obtained value.)

2.2 Gain Error

The gain error is the error in the slope of the transfer curve. For an ideal ADC, the gain is equal to 1 and the code count for any non-extreme code is equal to H_{ideal}. The ratio between the count of any non-extreme code and the ideal count H_{ideal} determines the actual ADC gain at that code. If the code count for a given code is smaller than H_{ideal}, the actual ADC gain at that code is greater than the ideal gain, and vice versa. While the measured value varies from code to code, ADC gain estimation is determined by averaging over m central codes of the range of the ADC

$$\text{Gain}^{-1} = \frac{\sum_{i=N_1}^{N_2} H(i)}{mH_{\text{ideal}}}. \tag{2}$$

2.3 Non-Linearity

Non-linearity can be presented by two different measures: differential non-linearity (DNL) and integral non-linearity (INL).

DNL measures the relative deviation of each code width from the ideal value (1 LSB). Given the histogram data, DNL is defined as the relative difference between the measured and the ideal code counts

$$\text{DNL}(i) = \frac{H(i) - H_{\text{ideal}}}{H_{\text{ideal}}}. \tag{3}$$

The INL determines the deviation of each output code center from the ideal straight center line. The INL for code i is computed from the cumulative sum of DNLs of the previous codes

$$\text{INL}(i) = \sum_{j=1}^{i} \text{DNL}(j). \tag{4}$$

The above equations include the operations of addition, subtraction, and division. While addition and subtraction are fairly simple operations, hardware implementation of division is more demanding. However, if the denominator is the power of 2, the division can be performed as shifting of the decimal point. In our case denominators are H_{ideal}, $2H_{\text{ideal}}$, and mH_{ideal}. In order to simplify the division, the H_{ideal} and m should be the power of 2. For subsequent explanation assume that $H_{\text{ideal}} = 2^P$, where P is some suitable integer value.

3 BIST STRUCTURES

The concept of the implemented histogram based BIST in a IEEE Std 1500 wrapper is shown in Figure 3(a). We use triangle-wave input signal, which in an ideal case results in a constant code count for non-extreme codes and equal code count for extreme codes.

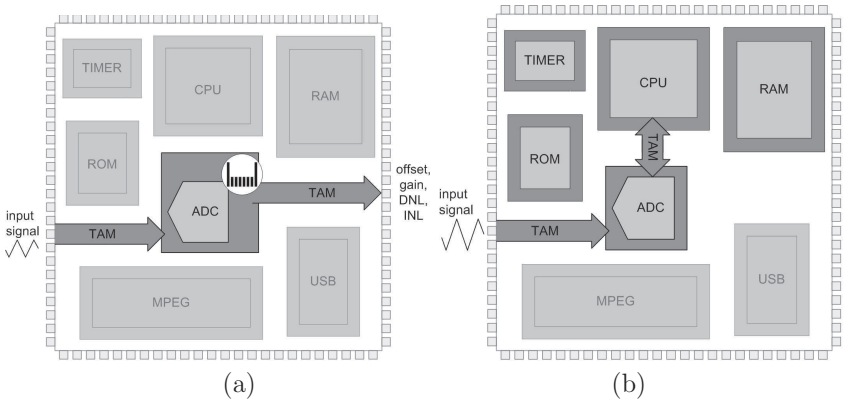


Fig. 3. The concept of (a) implemented BIST in IEEE Std 1500 test wrapper, (b) processor-based BIST

We first implemented two different BIST structures in a IEEE Std 1500 wrapper targeted either at low hardware overhead or at minimum test time. The first solution performs measurements and computations of static parameters in a sequence of individual steps. The ADC output data is processed on the fly in each step and the results (i.e., Offset, Gain, maximum value of DNL(i), maximum value of INL(i)) are transmitted out via test access mechanism. The second solution first collects the complete ADC test responses and stores them in a RAM. Next, the computation of static parameters is performed and the results are transmitted via TAM as in the previous case. We denote the first solution by sequential BIST and the second by RAM-based BIST.

In addition, another solution (shown in Figure 3(b)) was conceived in which a processor core performs ADC core test and computes the required test parameters.

3.1 Sequential BIST Approach

The sequential BIST structure depicted in Figure 4 operates in accordance with the concept proposed in [2]. It was, however, a bit modified in order to completely automate the evaluation of the DNL and INL parameters. The BIST structure is composed of three basic blocks:

- detector module, which monitors the codes generated by the ADC and signals when the code is equal to the preset value,
- exploitation module, which receives the signals from the detector module and performs the required operations (counting, complementing, etc.),
- control module, which coordinates the processing of both modules and connects the BIST structure to the IEEE Std 1500 test infrastructure. The control module is a state machine which generates the corresponding control signals of the

detector module and exploitation module. It also coordinates the operation of BIST and the IEEE Std 1500 test wrapper logic.

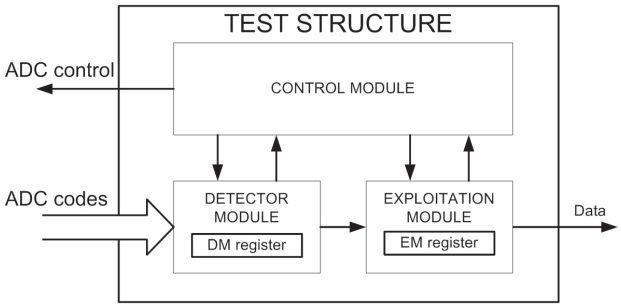


Fig. 4. Sequential BIST structure

The detector module contains a counter and a comparator. The counter is used for setting the required preset value which is stored in the DM register. The contents of the DM register is compared with the ADC code. If the ADC code equals the contents of the DM register, exploitation module is triggered. The exploitation module is an up/down counter with clear. In addition it is capable to calculate the complement of the current value. The temporary result is stored in the EM register.

The calculations of offset, gain, DNL and INL are separate processes performed in strictly sequential manner. While the sequential BIST structure described above requires low hardware overhead, the test time may become excessive. This is due to the fact that sequential approach takes N samples for the calculation of Offset, mN samples for the calculation of Gain, $(2^n - 2)N$ samples for the calculation of DNL and $(2^n - 2)N$ samples for the calculation of INL. As an alternative, RAM-based BIST has been developed.

3.2 RAM-Based BIST Approach

The RAM-based BIST test structure shown in Figure 5 consists of three components, which play a similar role as their counterparts in sequential BIST. The three components are:

- RAM module, in which the ADC code count of a complete histogram is stored,
- computation module, which computes the ADC static characteristics from the values stored in RAM,
- control module, which (similar to the sequential BIST) coordinates the processing of both blocks and connects the BIST structure to the IEEE Std 1500 test infrastructure.

In contrast to sequential BIST approach, only N samples are taken for the computation of static parameters of ADC. The samples are first stored in RAM, then

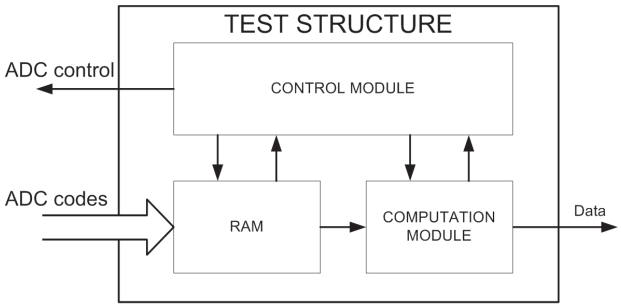


Fig. 5. RAM-based BIST structure

the static parameters are computed in accordance with the expressions 1–4. Some programming tricks are used in order to simplify the computation. For example, subtraction in expression 3 can be eliminated if the counts are first initialized to $-H_{\text{ideal}}$ and then increased by the number of occurrences of the code. RAM is organized in such way that the ADC code actually presents the RAM address of its code count. In this way, data manipulation is simplified.

3.3 Processor-Based BIST

Processor-based BIST exploits available microprocessor and other cores on the SoC for the execution of the ADC test. The idea is to reuse already present resources in the SoC and their interconnections to perform functional test of the ADC core. The microprocessor with timer interactions controls the ADC core, gathers the responses and stores them in the SoC RAM. After the stimulation period the gathered responses are evaluated by the processor. In a sense the processor with timer performs the role of the control and computation module. While this approach does not introduce any notable hardware overhead in the digital part of the SoC, additional ROM storage is required for the BIST program storage. In order to perform ADC BIST at full speed, which is generally faster then normal SoC operation, the processor in collaboration with other cores must be able to respond in time to the incoming ADC codes.

4 INTEGRITY CHECK OF THE TEST STRUCTURE

Integrity of the test infrastructure must be assured before we can trust the results of the implemented histogram test. For this purpose, integrity check is performed by replacing the tested ADC by a counter as shown in Figure 6. The counter generates codes corresponding to the ideal ADC with ideal triangular input waveform and the resulting static parameters are checked (Offset = 0, Gain = 1, DNL = 0, INL = 0). The two empty boxes in Figure 6 represent either detector and exploitation

module, or RAM and computation module since the same integrity check is used for sequential or RAM-based BIST.

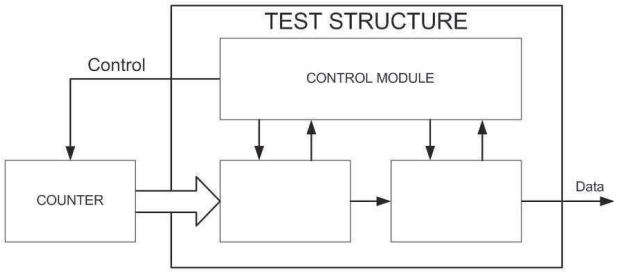


Fig. 6. Integrity check of a test structure

In processor-based BIST, integrity check of the test infrastructure includes testing of processor, timer and RAM cores, test of interconnects and test of ROM with stored BIST program. Since most of the above activities are included in general SoC test, only the test of ROM (i.e., checksum of its contents) is required in addition.

5 IMPLEMENTATION ISSUES

IEEE Std 1500 test wrapper with the above BIST structures was implemented in a Spartan3 XC3S200 FPGA. ADC0808 (an 8-bit ADC) was initially chosen as the unit-under-test. Measured parameters of the employed ADC were in conformance with the specifications provided by the manufacturer which demonstrates the feasibility of the histogram based test technique by the implemented test structures. Let us now compare the three solutions from some different points that may serve as the basis for selection of suitable strategy for BIST implementation of the embedded core.

The measurement uncertainty Δ (expressed in terms of LSB) is defined by

$$\Delta = \frac{1}{H_{\text{ideal}}}. \tag{5}$$

In sequential BIST approach, the number of samples required for processing a single ADC code is

$$N \approx 2^n H_{\text{ideal}}, \tag{6}$$

where n is the number of ADC bits. In our case, test signal was generated such that the number of occurrences of the two extreme codes was equal to the number of occurrences of the non-extreme codes, which gives

$$N = 2^n H_{\text{ideal}}. \tag{7}$$

The number of samples needed for the complete measurement is

$$N_{\text{all}} = (1 + m + 2^{n+1} - 4)N, \tag{8}$$

where m is the number of codes on which the gain calculation is performed.

In RAM-based BIST and processor-based BIST approach all codes are processed simultaneously. The number of samples required for the complete measurement is

$$N_{\text{all}} = N = 2^n H_{\text{ideal}}. \tag{9}$$

Table 1 presents measurement uncertainty Δ and overall test time of the three approaches for different values of H_{ideal} at 5 μs sample time.

	measurement	Test time [s] for 8-bit ADC		
	uncertainty	sequential	RAM-based	processor-based
H_{ideal}	Δ [LSB]	BIST	BIST	BIST
4	0.250	3.37	0.725	0.729
8	0.125	6.08	0.732	0.838
16	0.063	11.49	0.745	0.867
32	0.031	22.33	0.772	0.990
64	0.016	43.99	0.825	1.362

Table 1. Measurement uncertainty and overall test time of sequential, RAM-based and processor-based BIST

Notice that the test time in the case of sequential BIST is much higher than that of RAM-based BIST and processor-based BIST. While the test time of the 8-bit ADC at measurement uncertainty of 0.250 [LSB] may still be acceptable, the situation gets worse when testing ADC cores higher than 8-bit. Table 2 shows the test time of sequential, RAM-based and processor-based BIST for 8, 10, 12 and 16-bit ADC. The test times of sequential BIST of 12 and 16-bit ADCs are obviously not acceptable in practice.

	Test time [s] ($H_{\text{ideal}} = 4$)		
number of	sequential	RAM-based	processor-based
ADC bits	BIST	BIST	BIST
8	3.37	0.725	0.729
10	44.18	0.907	0.922
12	697.69	1.161	1.278
16	178,543.32	3.116	3.222

Table 2. Test time of sequential, RAM-based and processor-based BIST for different ADCs

RAM-based BIST and processor-based BIST are much faster but there are other issues that should be considered. As regards hardware overhead, RAM-based BIST employs much more resources than sequential BIST. We synthesized both approaches with Cadence RTL Compiler and the resulting designs are shown below. In order to get a realistic view of hardware overhead of individual solutions, sequential

number of ADC bits	sequential BIST			RAM-based BIST		
	gates	flip-flops	RAM cells	gates	flip-flops	RAM cells
8	881	233	–	461	156	2,560
10	911	249	–	541	172	12,288
12	961	265	–	583	188	57,344
16	1067	297	–	684	220	1,179,648

Table 3. Sequential and RAM-based BIST logic implementation (without IEEE Std 1500 wrapper logic)

and RAM-based BIST logic implementation without IEEE Std 1500 wrapper logic are summarized in Table 3.

Next, the resources of complete BIST implementations including wrapper control logic of the three solutions are summarized in Table 4.

number of ADC bits	sequential BIST			RAM-based BIST			processor-based BIST		
	gates	flip-flops	RAM cells	gates	flip-flops	RAM cells	gates	flip-flops	RAM cells
8	1012	293	–	594	216	2,560	63	59	–
10	1063	319	–	612	242	12,288	71	72	–
12	1131	345	–	680	268	57,344	97	95	–
16	1208	371	–	889	320	1,179,648	205	127	–

Table 4. BIST implementations including IEEE Std 1500 wrapper control logic

Processor-based BIST has minimal hardware overhead since the majority of its tasks are performed by the cores already included in a SoC. However, such a solution is specific for the given processor core and is not easily portable to other SoCs. On the other hand, sequential BIST and RAM-based BIST are autonomous and can be ported together with ADC core to other designs.

Scalability of the above solutions manifest itself in different issues. Analysis of BIST solutions for n-bit ADC cores shows that by increasing n the hardware overhead of sequential BIST increases linearly, exhibits exponential growth in the case of RAM-based BIST and practically does not change in processor-based BIST. Test time of sequential BIST grows exponentially while practically does not change in the case of RAM-based or processor-based BIST.

Providing high-quality stimulus signals is crucial to accomplish efficient and consistent BIST of a mixed-signal core. The signal can be internally generated in the test wrapper, or supplied via TAM either from an external signal generator or generated by another core (if available) in the SoC-under-test. In our case, including a triangle-wave generator [3] in the test wrapper would considerably increase the complexity hence externally supplied stimulus is a preferred approach. The impact of MOS switches connecting the external source to the ADC core was simulated by introducing an ABM module of a IEEE Std 1149.4 compliant test chip. For this purpose, our proprietary test chip [17] was employed. In order to assess the impact of MOS switches of analog boundary module on static parameter measurements, direct

connection to the ADC input was optionally provided. The measured parameters were in good agreement with the values obtained by the conventional technique using laboratory measurement equipment which gives confidence to the described approach.

6 CONCLUSIONS

The case study demonstrated that the three BIST solutions considerably differ in required resources, achieved test time, portability and scalability. Selection of the best test approach in practice depends on strategic decision taken in the early user requirement specification phase and is, of course, specific for each mixed-signal core of a SoC. Although the described implementation focused on a specific test technique (histogram based test) for a specific mixed-signal core (ADC) the three approaches can be generalized to other types of mixed-signal cores as follows:

- test wrapper performs core test and evaluates the results on the fly,
- test results (i.e., measurements) are stored in RAM and processed in batch,
- other cores of the SoC (if available) take the role of testing.

In purely digital SoC, the first two approaches are preferred since in most cases stimuli are generated by a LFSR (linear-feedback shift register) logic and test results are collected by a MISR (multiple input signature register) based on the same principles. This logic is compact and scalable and provides portability of design.

BIST of a SoC with mixed-signal cores faces different problems. Test methods possibly implemented in a test wrapper require specific logic which differs considerably from one case to another. There are no common parts, everything must be conceived from scratch. In fact, since a mixed-signal core is put in a SoC with well defined target application it is prudent to consider possible reuse of the application software (i.e., device drivers) in a processor-based BIST.

Processor-based BIST is traditionally employed in testing embedded memories. Its application in mixed-signal core test has not been widely investigated in literature although proprietary solutions in practice no doubt exist. In this regard, the comparison of empirical data presented in the paper allows an assessment of pros and cons of the two alternative approaches and may serve as a guideline in similar applications.

Generation of test stimuli is another issue in which BIST of mixed-signal cores differs from BIST of digital cores. While test pattern generation with LSFR is a common and widely practiced technique in digital word, analog stimulus generation [20] is a problem with no unique solution. Test signal generator included in the test wrapper results in increased complexity and hardware overhead. Besides, integrity check of a signal generator embedded in a test wrapper poses additional problems and increases test time. A simple alternative solution employing external signal generator is more likely to be applied in practice. In some cases, another core of

a SoC might take the role of stimulus generator, like for example oscillation-based test structures reported in [19].

At this time, IEEE Std 1500 test wrapper is defined only for digital cores. Its possible future extension to mixed-signal cores will probably follow the way of IEEE Std 1149.1 extended to IEEE Std 1149.4. The impact of MOS switches connecting the input signals (either from external source or from another core in a SoC) to the mixed-signal core will remain an important issue in practice.

REFERENCES

[1] Azaïs, F.—Bernard, S.—Bertrand, Y.—Renovell, M.: Towards an ADC BIST scheme using the histogram test technique, In Proceedings of the IEEE European Test Workshop Proceedings, pp. 53–58, 2000.

[2] Azaïs, F.—Bernard, S.—Bertrand, Y.—Renovell, M.: Implementation of linear histogram BIST for ADCs, In Proceedings of the Design Automation & Test in Europe Conference, pp. 590–595, 2001.

[3] Bernard, S.—Azaïs, F.—Bertrand, Y.—Renovell, M.: On-Chip Generation of Ramp and Triangle-Wave Stimuli for ADC BIST, Journal of Electronic Testing, Vol. 19, 2003, No. 4, pp. 469–479.

[4] Blair, J.: Histogram measurement of ADC nonlinearities using sine waves, IEEE Transactions on Instrumentation and Measurement, Vol. 43, 1994, No. 3, pp. 373–383.

[5] Carbone, P.—Chiorboli, G.: ADC sinewave histogram testing with quasi-coherent sampling, IEEE Transactions on Instrumentation and Measurement, Vol. 50, 2001, No. 4, pp. 949–953.

[6] Corrêa Alegria, F.—Arpaia, P.—Daponte, P.—Cruz Serra, A.: An ADC histogram test based on small-amplitude waves, Measurement, Vol. 31, 2002, No. 4, pp. 271–279.

[7] Corrêa Alegria, F.—Cruz Serra, A.: Error in the estimation of transition voltages with the standard histogram test of ADCs, Measurement, Vol. 35, 2004, No. 4, pp. 389–397.

[8] Corrêa Alegria, F.—Cruz Serra, A.: Overdrive in the standard histogram test of ADCs, Measurement, Vol. 35, 2004, No. 4, pp. 381–387.

[9] Corrêa Alegria, F.—Cruz Serra, A.: Standard Histogram Test Precision of ADC Gain and Offset Error Estimation, IEEE Transactions on Instrumentation and Measurement, Vol. 56, 2007, No. 5, pp. 1527–1531.

[10] Cruz Serra, A.—Corrêa Alegria, F.—Martins, R.—Fonseca da Silva, M.: Analog-to-digital converter testing–new proposals, Computer Standards & Interfaces, Vol. 26, 2004, No. 1, pp. 3–13.

[11] Dallet, D.—Machado da Silva, J.: Dynamic Characterisation of Analogue-to-Digital Converters, Springer, 2005.

[12] Doemberg, J.—Lee, H.-S.-Hodges, D. A.: Full-speed testing of A/D converters, IEEE Journal of Solid State Circuits, Vol. 19, 1984, No. 6, pp. 820–827.

[13] Hsin-Wen, T.—Bin-Da, L.—Soon-Jyh, C.: Histogram Based Testing Strategy for ADC, In Proceedings of the 15th Asian Test Symposium, pp. 51–54, 2006.

[14] Huertas, J. L.: Test and design-for-testability in mixed-signal integrated circuits, Kluwer Academic Publishers, 2004.

[15] IEEE Std 1500 - Standard for Embedded Core Test. Availaible on: http://grouper.ieee.org/groups/1500/, 2009.

[16] Mahoney, M.: DSP-based testing of analog and mixed-signal circuits, IEEE Computer Society Press, 1987.

[17] Kač, U.—Novak, F.—Azaïs, F.—Nouet, P.—Renovell, M.: Extending IEEE Std. 1149.4 Analog Boundary Modules to Enhance Mixed-Signal Test, IEEE Design & Test of Computers, Vol. 20, 2003, No. 2, pp. 32–39.

[18] Nadeaut-Dostie, B.—Adham, S. M. I.—Abbott, R.: Improved core isolation and access for hierarchical embedded test, IEEE Design & Test of Computers, Vol. 26, 2009, No. 1, pp. 18–25.

[19] Novak, F.—Santo Zarnik, M.—Kač, U.: Stimulus generation of a built-in-self-test using oscillation based test structures, International Journal of Electronics, Vol. 89, 2002, No. 11, pp. 811–820.

[20] Renovell, M.—Azaïs, F.—Bernard, S.—Bertrand, Y.: Hardware resource minimization for a histogram-based ADC BIST, In Proceedings of the 18th IEEE VLSI Test Symposium Proceedings, pp. 247–252, 2000.

[21] Roberts, G. W.—Lu, A. K.: Analog Signal Generation for Built-in Self-test of Mixed Signal Integrated Circuits, Kluwer Academic Publishers, 1995.

[22] Sandeep, K. G.—Marinissen, E. J.: A Test Time Reduction Algorithm for Test Architecture Design for Core-Based System Chips, Journal of Electronic Testing, Vol. 19, 2003, No. 4, pp. 425–435.

[23] Sehgal, A. E. J.—Sandeep, K. G.—Marinissen, E. J.—Chakrabarty, K.: Hierarchy-aware and area-efficient test infrastructure design for core-based system chips, In Proceedings of the Design Automation & Test in Europe Conference, pp. 285–290, 2006.

[24] Sinanoglu, O.—Marinissen, E. J.: Analysis of the Test Data Volume Reduction Benefit of Modular SOC Testing, In Proceedings of the Design Automation & Test in Europe Conference, pp. 182–187, 2008.

[25] Vanden Bossche, M.—Schoukens, J.—Renneboog, J.: Dynamic testing and diagnostics of A/D converters, IEEE Transactions on Circuits and Systems, Vol. 33, 1986, No. 8, pp. 775–785.

[26] Wagdy, M. F.—Awad, S. S.: Determining ADC effective number of bits via histogram testing, IEEE Transactions on Instrumentation and Measurement, Vol. 40, 1991, No. 4, pp. 770-772.

[27] Yong-Sheng, W.—Jin-Xiang, W.—Feng-Chang, L.—Yi-Zheng, Y.: Optimal Schemes for ADC BIST Based on Histogram, In Proceedings of the 14th Asian Test Symposium, pp. 52–57, 2005.

[28] Zivkovic, V.—Schat, J.—van der Heyden, F.—Seuren, G.: Core-based testing of embedded mixed-signal modules in a SoC, IEEE Design & Test of Computers, Vol. 26, 2009, No. 3, pp. 78–86.

Measurement Accuracy of Oscillation-Based Test of Analog-to-Digital Converters

Peter Mrak, Anton Biasizzo, and Franc Novak

Oscillation-based testing of analog-to-digital converters represents a viable option for low-cost built-in self-testing in mixed-signal design. While numerous papers have addressed implementation issues, little attention has been paid to the measurement accuracy. In this letter, we highlight an inherent measurement uncertainty which has to be considered when deriving the parameters from the oscillation frequency.

Keywords: mixed-signal test, analog-to-digital converter, (ADC) oscillation-based test (OBT), measurement accuracy.

I. Introduction

In the oscillation-based test (OBT) proposed in [1], a circuit-under-test is transformed into an oscillator, and the frequency of oscillation is compared to a reference value obtained from a known-good circuit operating in the same measurement conditions. Discrepancy between the oscillation frequency of a circuit-under-test and the reference value indicates possible faults.

The OBT technique has been applied to various kinds of mixed-signal circuits [2]-[8].

In OBT of an analog-to-digital converter (ADC), a feedback loop is provided which forces the ADC to oscillate around a selected code *i* [7]. The feedback loop is fed to a circuit which generates a triangle wave signal of symmetrical slope as shown in Fig 1. The slope of the triangle wave signal is inverted one sample time after the sampled input signal attains or crosses the threshold voltage as shown in Fig. 2.

The measured frequency of the triangle wave signal can serve as the basis of a go/no-go ADC test. Furthermore, it can be used to determine static ADC parameters such as differential nonlinearity (DNL) and integral nonlinearity (INL). While other authors have studied implementation issues of OBT, such as input signal generation, digital comparators, and feedback circuitry, none of them have addressed the measurement accuracy of the test method.

In this letter, we focus on the mechanism of generating the triangle waveform signal and demonstrate an inherent measurement uncertainty which has to be considered when deriving the parameters from the oscillation frequency.

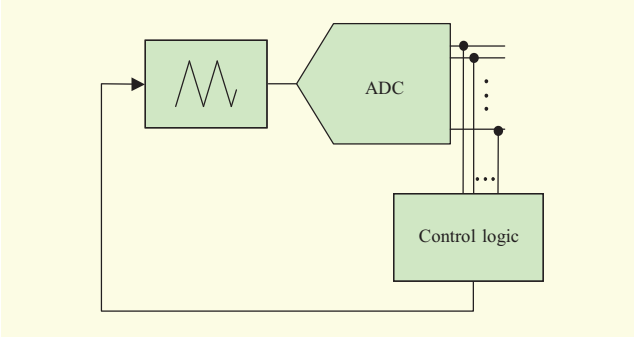


Fig. 1. Oscillation-based test structure of ADC.

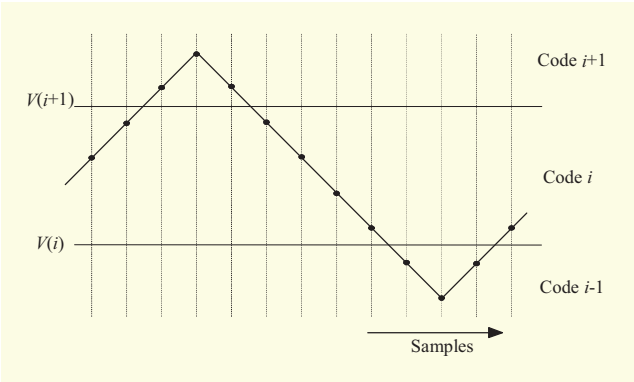


Fig. 2. Sampling of triangle wave input signal.

Manuscript received July 13, 2009; revised Oct. 8, 2009; accepted Oct. 19, 2009.

Peter Mrak (phone: +386 1 4773 386, email: peter.mrak@ijs.si), Anton Biasizzo (email: anton.biasizzo@ijs.si), and Franc Novak (corresponding author, email: franc.novak@ijs.si) are with the Computer Systems Department, Jozef Stefan Institute, Ljubljana, Slovenia.

doi:10.4218/etrij.10.0209.0285

II. Preliminaries

An ADC generates a single output code for a range of input voltages. The voltage level where the ADC output changes to code i is defined as the threshold voltage $V(i)$. The difference between the threshold voltages $V(i+1)$ and $V(i)$ is called the code width $Q(i)$. The code widths of an n-bit ideal ADC are equal and are determined by dividing the full scale range (FSR) by the number of ADC codes:

$$Q_{\mathrm{N}}(i) = Q_{\mathrm{N}} = \frac{FSR}{2^n}. \tag{1}$$

DNL is a figure of merit that describes the uniformity of steps between ADC codes. The deviation of the threshold voltages from their nominal values causes different sizes of code widths. DNL for a selected code i is given by

$$DNL(i) = \frac{Q(i) - Q_{\mathrm{N}}}{Q_{\mathrm{N}}}. \tag{2}$$

Without loss of generality, we assume that the slope S of the triangle wave signal is generated such that the signal traverses the nominal code width Q_{N} in a whole number k_{N} of sample periods T_S:

$$Q_{\mathrm{N}} = S \cdot k_{\mathrm{N}} \cdot T_S. \tag{3}$$

The actual code width $Q(i)$ is obtained by determining the time $k \cdot T_S$ of the input signal passing between $V(i)$ and $V(i+1)$:

$$Q(i) = S \cdot k \cdot T_S. \tag{4}$$

Evaluation of k from the measured oscillation frequency represents the basis of our further discussion.

III. Evaluation of Measurement Accuracy

The conversion of an ADC is not instantaneous; the acquired code is delayed for the conversion time. The output of the ADC is compared to the selected code i at the sampling instances, and the slope of the triangle signal is inverted depending on the result. The triangle wave signal is sampled periodically at $T(j)$ as shown in Fig. 3.

Let us denote the first sample time as $T(-1)$, where the input signal is equal to or greater than the threshold voltage $V(i+1)$. Since the output code of the ADC is delayed for the sample time T_S, the slope of the input signal is inverted at the sample time $T(0)$. In general, the transition of the input signal over the threshold voltage occurs at $T(-1)$ or somewhere within the interval between $T(-2)$ and $T(-1)$. For further analysis, let us denote the time between the transition of the threshold voltage $V(i+1)$ and the sample time $T(-1)$ as $\delta \cdot T_S$, where the values of δ are

$$0 \le \delta < 1. \tag{5}$$

The time from the transition of the threshold voltage $V(i+1)$ to the inversion of the slope at $T(0)$ is $(1+\delta) \cdot T_S$. Since the rising and the falling slopes are equal, the transition of the threshold voltage $V(i)$ occurs at $(1+\delta+k) \cdot T_S$ after $T(0)$. Depending on the values of k and δ, we distinguish two different cases:

$$k + \delta < \lceil k \rceil, \tag{6}$$

$$k + \delta \ge \lceil k \rceil, \tag{7}$$

where $\lceil\ \rceil$ is the ceiling operator which rounds a number to its nearest integer value in an upwards direction. This results in two different signals, which are represented in Fig. 3 by the continuous and the dashed lines. The first case refers to the situation in which the signal transition of the threshold voltage $V(i)$ occurs before $T(\lceil k \rceil + 1)$, while the second case refers to the situation in which the signal transition of the threshold voltage $V(i)$ occurs after $T(\lceil k \rceil + 1)$. Consequently, the inversion of the slope of the two signals differs for one sample time, which leads to two different oscillation periods:

$$T_1 = (2 \cdot \lceil k \rceil + 4) \cdot T_S, \tag{8}$$

$$T_2 = (2 \cdot \lceil k \rceil + 6) \cdot T_S. \tag{9}$$

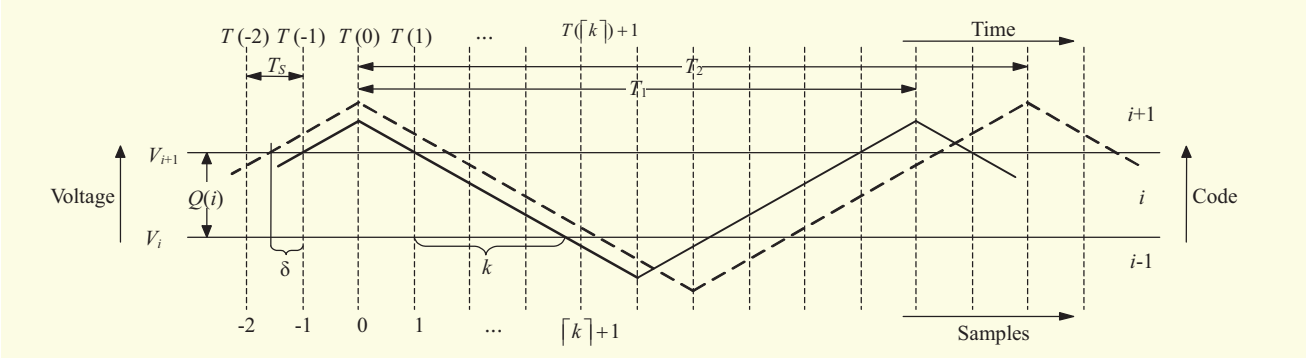


Fig. 3. Detailed timing diagram.

Table 1. Simulated results for $DNL(2) = 0.3$.

δ / k_N	Measured $DNL(2)$								Meas. uncert.
	0.062	0.188	0.312	0.438	0.562	0.688	0.812	0.938	
2	0.00	0.00	0.00	0.50	0.50	0.50	0.50	0.50	0.500
3	0.00	0.33	0.33	0.33	0.33	0.33	0.33	0.33	0.333
4	0.25	0.25	0.25	0.25	0.25	0.25	0.50	0.50	0.250
6	0.17	0.17	0.33	0.33	0.33	0.33	0.33	0.33	0.167
8	0.25	0.25	0.25	0.25	0.25	0.38	0.38	0.38	0.125
16	0.25	0.25	0.31	0.31	0.31	0.31	0.31	0.31	0.062
32	0.28	0.28	0.28	0.31	0.31	0.31	0.31	0.31	0.031

For an ideal ADC, the input signal traverses the code width in a whole number k_N of sample periods T_S as assumed in (3). Since k_N is a whole number, the first of the two considered cases described by (6) and (7) is not feasible. The only oscillation period, denoted by T_N, is thus

$$T_N = (2 \cdot k_N + 6) \cdot T_S \,. \tag{10}$$

The actual measured oscillation period T is either T_1 or T_2. Because we cannot distinguish between the two cases, the computation of k from expressions (8) and (9) gives

$$k = \frac{T}{2 \cdot T_S} - 3 \pm 1 \,. \tag{11}$$

The two possible outcomes are reflected in the code width $Q(i)$:

$$Q(i) = S \cdot \left(\frac{T}{2 \cdot T_S} - 3 \right) \cdot T_S \pm S \cdot T_S \,. \tag{12}$$

From the definition of DNL we obtain

$$DNL(i) = \frac{T - T_N}{2 \cdot k_N \cdot T_S} \pm \frac{1}{k_N} \tag{13}$$

for the selected code i.

For illustration, an 8-bit ADC with $DNL(2) = 0.3$ was considered. The OBT technique was modeled in the Matlab Simulink environment. We simulated the OBT for different values of k_N and δ. Obtained values of $DNL(2)$ and the corresponding measurement uncertainty are given in Table 1.

IV. Conclusion

An oscillation-based test of ADC manifests in two different oscillation periods depending on the time of the transition of the threshold voltage relative to the sampling. Since we cannot distinguish between the two cases, the ambiguity of the frequency of oscillation should be considered in OBT implementations. We demonstrated that the measurement uncertainty of the DNL due to this ambiguity is $1/k_N$.

References

[1] K. Arabi and B. Kaminska "Oscillation-Test Strategy for Analog and Mixed-Signal Circuits," *Proc. VLSI Test Symp.*, 1996, pp. 476-482.

[2] K. Arabi and B. Kaminska, "Design for Testability of Integrated Operational Amplifiers Using Oscillation-Test Strategy," *Proc. Int. Conf. Comput. Design*, 1996, pp. 40-45.

[3] K. Arabi and B. Kaminska, "Testing Analog and Mixed-Signal Integrated Circuits Using Oscillation-Test Method," *IEEE Trans. Computer-Aided Design Integr. Circuits Syst.*, vol. 16, no. 7, 1997, pp. 745-753.

[4] G. Huertas et al., "Effective Oscillation-Based Test for Application to a DTMF Filter Bank," *Proc. Int. Test Conf.*, 1999, pp. 549-555.

[5] M. Santo Zarnik, F. Novak, and S. Macek, "Design of Oscillation-Based Test Structures for Active RC Filters," *IEE Proc. Circuits Devices Syst.*, vol. 147, no. 5, 2000, pp. 297-302.

[6] P.M. Dias, J.E. Franca, and N. Paulino, "Oscillation Test Methodology for a Digitally-Programmable Switched-Current Biquad," *Proc. IEEE Int. Mixed Signal Testing Workshop*, 1996, pp. 221-226.

[7] K. Arabi and B. Kaminska, "Oscillation Built-In Self Test (OBIST) Scheme for Functional and Structural Testing of Analog and Mixed-Signal Integrated Circuits," *Proc. Int. Test Conf.*, 1997, pp. 786-795.

[8] G. Huertas et al., "Oscillation-Based Test in Bandpass Oversampled A/D Converters," *Microelectron. J.*, vol. 34, no. 10, 2003, pp. 927-936.

Generation of sequential diagnostic procedures including symmetrical, asymmetrical and multi-valued tests

P. Mrak[1], F. Novak[2], A. Biasizzo[3]

Abstract – *The paper deals with the problem of minimizing the cost of the sequence of tests required to localize a fault. The interactive sequential diagnosis tool supporting symmetrical, asymmetrical and multi-valued tests is presented. The main part of the paper is devoted to practical guidelines in the form of illustrative examples that may help the user to acquire and formalize the input data and to use the tool for the generation of optimal decision trees.*

Keywords*: diagnostic tools, fault location, system level diagnostics*

I. Introduction

Increasing complexity of electronic systems makes the problem of testing more and more difficult. Design-for-testability techniques [1] try to keep the problem complexity at the level that can be managed with the established testing techniques in practice. However, in production test or system maintenance, fault detection may not be sufficient. Faults must be isolated to determine the actual cause and origin of defects in order to modify the production process and/or repair the product. In system maintenance, detected failures must be diagnosed and repaired as rapidly as possible to return the system to the correct operation. Fault isolation usually requires more thorough and costly procedures than fault detection. It is therefore imperative to find low-cost solutions of the diagnosis problem.

System operation in the presence of faults can be presented by different system failure states. The goal of the diagnostic procedure is to identify the actual failure state by executing a sequence of tests which provide some information on system behavior. In principle, any measurement, signal, or other observable event can be viewed as a test.

Consider a test with a binary outcome (pass or fail) in such a diagnostic procedure. Assume that before the test is executed, the system is in a system state $s_i \epsilon S$, where S denotes the set of current candidate system states. The test splits the set S into subsets A and B. Let A denote the resulting set of candidate system states when the test fails and B the set of candidate system states when the test passes.

For a *symmetrical* test: $A \cap B = 0$. In other words, the test fails if and only if the system is in the faulty state $s_i \epsilon A$. Hence, if a symmetrical test passes, system states A are omitted from the set of the candidate faulty system states. A symmetrical test thus has 100% fault coverage for the set A of candidate faulty system states.

For an *asymmetrical* test: $A \cap B \neq 0$. An example of an asymmetrical test is the alarm indicator on a control panel. If the light is on, the actual system is one of the candidate faulty system states of A. On the other hand, if the light is not on, one cannot distinguish between the two cases: either there is no alarm or the indicator bulb is bad.

The notion of symmetrical and asymmetrical test can be associated with the definition of complete and incomplete test [2].

Determining the sequence of tests required to reach a diagnostic conclusion at minimum cost is known as the test sequencing problem [3]. Different algorithms have been proposed for its solution [4] - [7]. Traditional test sequencing problem has been defined for symmetrical and binary tests. In this paper we present an interactive software tool for the generation of sequential diagnosis procedures supporting also asymmetrical and multi-valued tests.

II. Generalized test sequencing problem

We generalize the traditional definition of test sequencing problem [8] by including asymmetrical and multi-valued tests. The generalized test sequencing problem is formally defined in the following way.

For a system described by:

- the set of system states $S = \{s_0, s_1, \ldots, s_m\}$ where s_0 denotes the fault-free state of the system and s_i, $(1 \leq i \leq m)$ denotes one of m potential faulty states of the system;
- the a-priori probabilities of system states $p(S) = [p(s_0), p(s_1), \ldots, p(s_m)]$;
- the set of available tests $T = \{t_1, t_2, \ldots, t_n\}$;
- test costs $c = [c_1, c_2, \ldots, c_n]$ associated with the set of tests T;
- the test outcomes: $R = \{R_1, R_2, \ldots, R_n\}$; Let $Rj = \{r_1, \ldots, r_d\}$ present the possible outcomes of a test t_j. With a pair (s_i, t_j) we associate the corresponding

Manuscript received January 2007, revised January 2007

test outcome r_k if test t_j is symmetrical for system state s_i. In the case where test t_j exhibits asymmetrical property for a given system state s_i, the corresponding test outcomes are given together with their conditional probabilities {(r_{k1}, $p(r_{k1})$),(r_{k2}, $p(r_{k2})$)…}. (Notice that the sum of the probabilities of the test outcomes is 1.)

the problem is to generate the diagnostic procedure such that the criterion function given by

$$J = \sum_{i=0}^{m} \sum_{j=1}^{n} \alpha_{ij}\, p(s_i) c_j \qquad (1)$$

(i.e., the average cost of the decision tree) is minimized. In the above expression, $\alpha_{ij} = 1$ if test t_j is used in the path leading to the identification of system state si and is 0 otherwise.

The test sequencing problem can be formulated as a best-first-search on an AND/OR graph. The problem is known to be NP-complete [3] therefore a cost-to-go heuristic estimation is used to guide the AND/OR graph search. Our implementation of the optimal AND/OR graph search techniques corresponds to the algorithm AO*. The algorithm AO* is ordered best-first search algorithm; it expands only the node of the search graph that offers the most promising way of reaching the goal nodes on the basis of two heuristics: (i) the first derived by appealing to the analogy between the test sequencing and the Huffman coding problem (HEF1) and (ii) entropy-based heuristics (HEF2). Yet another one, (HEF3) is equal to HEF2 + 1. Using heuristics HEF1 or HEF2, AO* is guaranteed to find an optimal solution. The definition of heuristics and the proof of optimality of AO* is given in [8]. Decision between using HEF1 and HEF2 depends on the size of the problem. HEF1 is recommended for small and medium-sized problems ($m < 100$), while for larger problems HEF2 preferred. HEF3 does not always give an optimal solution but it is useful for complex examples which are not tractable with HEF1 and HEF2.

Sub-optimal algorithms provide trade-off between optimality and computational complexity. They perform local step-by-step optimization. The sequential diagnosis tool described in this paper offers two such algorithms: information heuristic and separation heuristic algorithm (also denoted as PQ algorithm).

The traditional definition of the test sequencing problem assumed only binary symmetrical tests. However, available diagnostic tests in practice may have an arbitrary number of possible outcomes (multi-valued test) and may be unreliable to a certain degree (asymmetrical test).

The question arose if the above algorithms could generate the solutions for the generalized case including multi-valued and asymmetrical tests. It has been proved that the same heuristics can be employed in the generalized case [8]. Consequently, the definition of the test sequencing problem given in this paper includes multi-valued and asymmetrical tests.

III. Diagnostic tool

A tool that generates solutions to the generalized test sequencing problem has been implemented and is available for free download at http://csd.ijs.si/applications/SDT/sdt.html. In this paper we describe how the tool can be used for system diagnosing and indicate possible applications in practice. Implemented web interface enables the user to run the tool remotely on JSI server. The purpose of the web interface is to give the potential user the idea of the features provided by the diagnostic tool.

A collection of examples covering different test variant situations is provided as a quick way to get acquainted with the tool. For each example, the diagnostic procedure can be generated by AO* algorithm (with HEF1, HEF2 and HEF3 heuristics) and PQ algorithm. Since the generation of diagnostic procedures is computationally intensive, the external users are urged to run the diagnostic tool on JSI server for reasonably small examples.

III.1. How to use the diagnostic tool

An illustrative example of a system with 9 faulty states is shown in Table 1. Tests t_1 and t_4 are symmetrical binary tests (with outcomes pass, fail). Test t_2 is symmetrical multi-valued test (with outcomes falling into measurement intervals r_0, r_1, r_2 and r_3). Test t_3 is multi-valued test (with test outcomes a, b, c and d). This test is asymmetrical for faulty states 3, 6 and 9. For example, in the presence of faulty system state 3, the probability of test outcome a is 0.6 and the probability of test outcome d is 0.4. The test costs vary from 0.7 to 1.7.

system state	tests				probability
	test 1	test 2	test 3	test 4	p(s)
fault_free	pass	measured_r0	a	pass	0.6
faulty_1	fail	measured_r2	b	pass	0.3
faulty_2	pass	measured_r1	b	pass	0.02
faulty_3	fail	measured_r0	a(0.6); d(0.4)	fail	0.02
faulty_4	pass	measured_r3	a	fail	0.01
faulty_5	pass	measured_r2	d	pass	0.01
faulty_6	fail	measured_r1	a(0.7); d(0.3)	pass	0.01
faulty_7	pass	measured_r2	c	fail	0.01
faulty_8	fail	measured_r0	c	pass	0.01
faulty_9	fail	measured_r1	b(0.9); c(0.1)	fail	0.01
test costs	1	1.1	1.7	0.7	

Table 1. Input data for generation of the decision tree for the first illustrative example

The diagnostic tree generated by the tool is shown in Figure 1. The average cost in this case is 2.139.

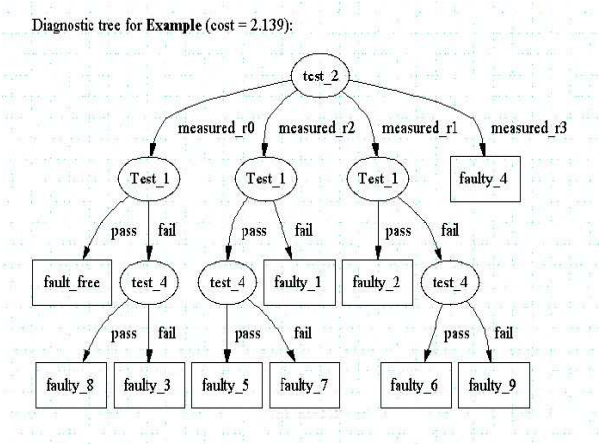


Fig. 1. Generated diagnostic tree for the first illustrative example

In the following we briefly sketch our current research in system diagnosis in order to show how the presented diagnosis tool can be used in practice.

We refer to the work of Murray [9] and the notion of test propagation through system modules. Typical system design scenario consists of different abstraction levels with physical level at the bottom and architecture level as the highest level. During a hierarchical system design, a collection of tests is generated for fault detection in individual system parts. The transparency analysis of these pre-computed tests is performed in the next step in order to organize them in an efficient overall system test [9]. While Murray actually addresses the problem of fault detection in a complex system, we extend the notion of transparency for the purpose of system diagnosis. While the original pre-computed tests may be sufficient for fault detection, serial chains of functional blocks make the problem of fault localization rather difficult. Modifications (i.e., extensions) of original sets of pre-computed tests are normally needed to achieve adequate controllability and observability for system diagnosis. In practice, different approaches can be taken, introducing for example additional lines for accessing internal points from primary inputs or outputs, or even adding a scan chain logic to the original design. Detailed description of this issue is beyond the scope of the paper. In the following we assume that this problem has been solved and we concentrate mainly on the selection of tests for system diagnosis. We analyze the transparency of individual system parts and derive solutions in accordance with available tests (i.e., test vectors generated at the outputs of preceding system parts). We use the diagnosis tool for generating optimal diagnostic trees of system parts. In order to get optimal solution at the system level we include the cost of transmitting the results to the primary system outputs in the cost of individual tests.

As an example of generating the diagnostic tree for a system part consider the computational element of radix-4 pipeline Cooley-Tukey FFT module shown in Figure 2. The diagnostic test data are presented in Table 2 (split into 3 parts).

The circuit consists of three multipliers (M1 - M3) and eight adders (A1 - A8), four performing addition and four performing subtraction. With a failure of individual functional block we associate a faulty system state (presented in the left-most column in the table). The probabilities of system states, including fault-free system state s0, are given in the right-most column.

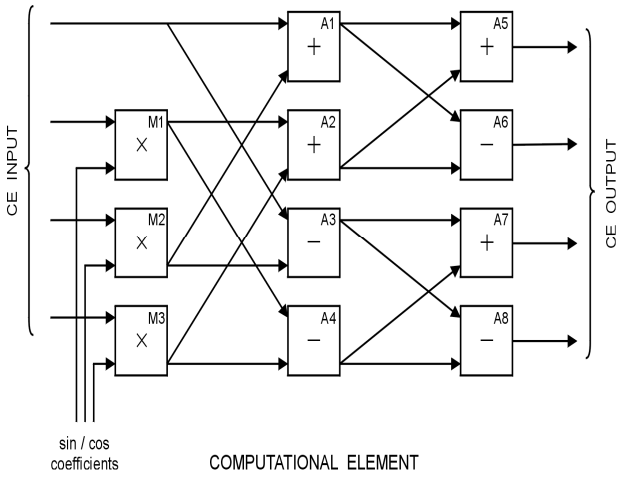


Fig. 2. Computational element as a part of radix-4 pipeline Cooley-Tukey FFT module

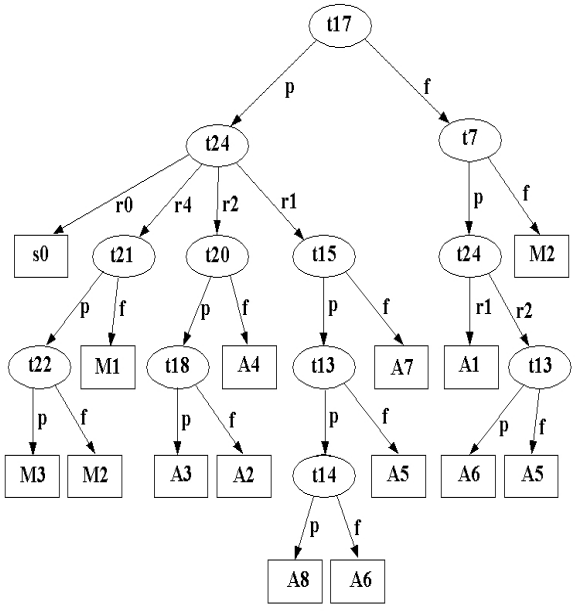


Fig. 3. The resulting minimum cost diagnostic tree of the computational element

state	t1	t2	t3	t4	t5	t6	t7	t8	prob
s0	p	p	p	p	p	p	p	p	0.02
M1	f	f	f	f	f(0.05) p(0.95)	f(0.05) p(0.95)	f(0.05) p(0.95)	f(0.05) p(0.95)	0.14
M2	f(0.05) p(0.95)	f(0.05) p(0.95)	f(0.05) p(0.95)	f(0.05) p(0.95)	f	f	f	f	0.14
M3	f(0.1) p(0.9)	f(0.1) p(0.9)	f(0.1) p(0.9)	f(0.1) p(0.9)	f(0.05) p(0.95)	f(0.05) p(0.95)	f(0.05) p(0.95)	f(0.05) p(0.95)	0.14
A1	f(0.25) p(0.75)	f(0.25) p(0.75)	p	p	f(0.8) p(0.2)	f(0.8) p(0.2)	p	p	0.07
A2	f(0.8) p(0.2)	f(0.8) p(0.2)	p	p	f(0.25) p(0.75)	f(0.25) p(0.75)	p	p	0.07
A3	p	p	f(0.25) p(0.75)	f(0.25) p(0.75)	p	p	f(0.8) p(0.2)	f(0.8) p(0.2)	0.07
A4	p	p	f(0.8) p(0.2)	f(0.8) p(0.2)	p	p	f(0.25) p(0.75)	f(0.25) p(0.75)	0.07
A5	f(0.5) p(0.5)	p	p	p	f(0.5) p(0.5)	p	p	p	0.07
A6	p	f(0.5) p(0.5)	p	p	p	f(0.5) p(0.5)	p	p	0.07
A7	p	p	f(0.5) p(0.5)	p	p	p	f(0.5) p(0.5)	p	0.07
A8	p	p	p	f(0.5) p(0.5)	p	p	p	f(0.5) p(0.5)	0.07
cost	2	2	2	2	2	2	2	2	

state	t9	t10	t11	t12	t13	t14	t15	t16	prob
s0	p	p	p	p	p	p	p	p	0.02
M1	f(0.1) p(0.9)	f(0.1) p(0.9)	f(0.1) p(0.9)	f(0.1) p(0.9)	f(0.1) p(0.9)	f(0.1) p(0.9)	f(0.1) p(0.9)	f(0.1) p(0.9)	0.14
M2	f(0.05) p(0.95)	f(0.05) p(0.95)	f(0.05) p(0.95)	f(0.05) p(0.95)	f(0.1) p(0.9)	f(0.1) p(0.9)	f(0.1) p(0.9)	f(0.1) p(0.9)	0.14
M3	f	f	f	f	f(0.1) p(0.9)	f(0.1) p(0.9)	f(0.1) p(0.9)	f(0.1) p(0.9)	0.14
A1	f(0.25) p(0.75)	f(0.25) p(0.75)	p	p	f(0.4) p(0.6)	f(0.4) p(0.6)	p	p	0.07
A2	f(0.8) p(0.2)	f(0.8) p(0.2)	p	p	f(0.4) p(0.6)	f(0.4) p(0.6)	p	p	0.07
A3	p	p	f(0.25) p(0.75)	f(0.25) p(0.75)	p	p	f(0.4) p(0.6)	f(0.4) p(0.6)	0.07
A4	p	p	f(0.8) p(0.2)	f(0.8) p(0.2)	p	p	f(0.4) p(0.6)	f(0.4) p(0.6)	0.07
A5	f(0.5) p(0.5)	p	p	p	f	p	p	p	0.07
A6	p	f(0.5) p(0.5)	p	p	p	f	p	p	0.07
A7	p	p	f(0.5) p(0.5)	p	p	p	f	p	0.07
A8	p	p	p	f(0.5) p(0.5)	p	p	p	f	0.07
cost	2	2	2	2	1	1	1	1	

state	t17	t18	t19	t20	t21	t22	t23	t24	prob
s0	p	p	p	p	p	p	p	e0	0.02
M1	p	f(0.2) p(0.8)	p	f(0.2) p(0.8)	f	p	p	e4	0.14
M2	f(0.2) p(0.8)	p	f(0.2) p(0.8)	p	p	f	p	e4	0.14
M3	p	f(0.2) p(0.8)	p	f(0.2) p(0.8)	p	p	f	e4	0.14
A1	f	p	p	p	p	f(0.8) p(0.2)	p	e2	0.07
A2	p	f	p	p	f(0.8) p(0.2)	p	f(0.8) p(0.2)	e2	0.07
A3	p	p	f	p	p	f(0.8) p(0.2)	p	e2	0.07
A4	p	p	p	f	f(0.8) p(0.2)	p	f(0.8) p(0.2)	e2	0.07
A5	f(0.4) p(0.6)	f(0.4) p(0.6)	p	p	f(0.3) p(0.7)	f(0.3) p(0.7)	f(0.3) p(0.7)	e1	0.07
A6	f(0.4) p(0.6)	f(0.4) p(0.6)	p	p	f(0.3) p(0.7)	f(0.3) p(0.7)	f(0.3) p(0.7)	e1	0.07
A7	p	p	f(0.4) p(0.6)	f(0.4) p(0.6)	f(0.3) p(0.7)	f(0.3) p(0.7)	f(0.3) p(0.7)	e1	0.07
A8	p	p	f(0.4) p(0.6)	f(0.4) p(0.6)	f(0.3) p(0.7)	f(0.3) p(0.7)	f(0.3) p(0.7)	e1	0.07
cost	3	3	3	3	8	8	8	10	

Table 2. The diagnostic test data of Computational element as a part of radix-4 pipeline Cooley- Tukey FFT module

In this particular case we have at disposal a collection of 24 tests with diagnostic capabilities described in the table. Tests t_1 - t_{23} are asymmetrical with binary outcome pass or fail. Each of them exhibits symmetrical behavior for some system states and asymmetrical for the others. For example, t_{19} exhibits symmetrical behavior for system states s_0, *M1, M3, A1, A2, A3, A4, A5, A6,* and asymmetrical for *M2, A7, A8*. In the user interface of the tool, asymmetrical behavior of a test is described by explicitly stating the probabilities of test outcomes for the corresponding system states. For example, for the system state *A7* the probability of test outcome fail of test t_{19} is 0.4 and probability of test outcome pass is 0.6. Test t_{24} is symmetrical multi-valued test with outcomes e0, e1, e2, e4. The bottom row of the table describes test costs. The resulting minimum cost diagnostic tree is shown in Figure 3.

As it can be seen from the above the *a priori* probability data of system states and the probability data of asymmetric test outcomes are needed for the generation of test sequences. They can be assessed or computed from MIL failure rate data tables. Another way to obtain this data is by collecting the diagnoses through the system life cycle. In this way, initially estimated probabilities are improving on reliability during system operation.

$$p_b = \left(\frac{n}{n+m}\right)\frac{r}{n} + \left(\frac{m}{n+m}\right)p_a \qquad (2)$$

The *m*-estimation approach can be used to combine initial estimation and the accumulated knowledge about system failures. The parameter *m* in the above equation is a weight, by which one can regulate the impact of the

initially estimated probabilities p_a on the updated probabilities p_b, and the impact of relative frequency r/n, with n trials and r successes.

III.2. Applications

The described sequential diagnostic tool has been used in connection with SATAN software (Systems Automatic Testability Analysis) [10]. SATAN package has been designed for testability analysis and for functional specification of test programs and is based on a model of information transfer of the unit-under-test (UUT). An information transfer model consists of a bipartite graph made up of places and transitions. Whereas transitions specify the different modes of information transfer between places, places correspond either to UUT inputs (sources), UUT outputs (destinations) or to modules. The modules make up the functions of information transformation or the functions of interconnection between components. After a comparative case study of SATAN and sequential diagnostic tool reported in [11], the complementary features of the two approaches were merged in an experimental diagnostic environment and used on experimental case studies in the area of software testing.

An earlier version of the sequential diagnosis tool [12] has also been used in individual student research projects in different areas of application such as medicine, system security, and others. The tool has been later upgraded by including the user interface with a collection of examples covering different test variant situations. The examples can be modified so that the user can tailor them according to his/her needs in practice.

Currently we are developing production test and field maintenance procedures for a cooling appliance in an industrial project in collaboration with Gorenje. While the UUT in this case is indeed a new product, the definition of faulty states and associated probabilities are based on the empirical data gathered from similar product types. The sequential diagnosis tool is used for the generation of optimal diagnostic procedures in terms of test execution time.

It can also be used in other applications, like mechanical system troubleshooting, generation of optimal emergency scenarios in safety critical applications, in medicine, etc. General strategies for system diagnosis are described in [13]. The tool supports the above diagnostic strategies and provides a suitable platform for experimental evaluation of alternative solutions in practice.

III.3. Limitations

A case study of diagnosing faults in sequential multi-valued logic networks has been performed. Obtained results summarized in Table 3 describe the performance of the tool and point to its limitations in practice. The test problems were generated using a uniform distribution. For simplicity all tests have cost equal to one. All experiments have been carried out on a system with AMD Athlon™ 64 Dual Core Processor 6000+ and 1.97 GByte of main memory. (All runtimes are given in CPU seconds, minutes and hours.)

States	Algorithm	Nodes	Back-tracks	CPU time
100	HEF_1	124	220	0.04
	PQ	136	0	0.01
250	HEF_1	616	84	0.29
	PQ	358	0	0.05
500	HEF_1	1025	297	0.91
	PQ	697	0	0.10
750	HEF_1	2057	112	1.55
	PQ	1351	0	0.19
1000	HEF_1	2400	1144	2.14
	PQ	1351	0	0.36
2500	HEF_1	7090	225	10.81
	PQ	3585	0	0.13
5000	HEF_1	12791	512	28.79
	PQ	6971	0	2.56
7500	HEF_1	32546	2938	1:39.91
	PQ	10119	0	4.04
10000	HEF_1	30877	512	2:05.72
	PQ	13831	0	11.49
15000	HEF_1	41330	513	3:11.74
	PQ	20384	0	17.84
20000	HEF_1	52784	577	6:18.03
	PQ	26717	0	29,6
25000	HEF_1	55775	549	7:38.35
	PQ	32476	0	30.04

Table 3. Performance of sequential diagnosis tool running AO* (HEF1) and PQ algorithm

IV. Future work

Generation of minimum cost diagnostic procedures is an important issue in complex systems in different fields such as the nuclear and chemical industry. In order to develop efficient solutions, a systematic framework for constructing useful diagnostic procedures is needed. An attempt in this direction is made in [14]. The described approach combines the decision-making strategies of human and the test sequencing technique. Recently, another approach based on integrating concepts from one-step look-ahead heuristic algorithms and basic ideas of Huffman coding to construct an AND/OR decision tree bottom-up has been proposed [15]. In our future version of the sequential diagnosis tool selected issues from the above work will be considered. In addition, we plan to expand the tool to cover multiple fault situations. Theoretical background can be found in [16], [17].

V. Conclusion

The aim of this contribution is to present the interactive software tool that provides solutions to the generalized test sequencing problem including asymmetrical and multi-valued tests. It is a general purpose diagnostic aid and can be used in different applications in practice. It can be employed as a kernel of a diagnostic reasoner in automatic test equipment or system maintenance.

Acknowledgements

This work was supported by the Slovenian Research Agency (ARRS) under grant P2-0098.

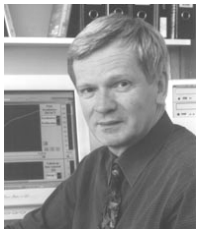
References

[1] M.Abramovici, M.A.Breuer, A.D.Friedman, *Digital Systems Testing and Testable Design* (IEEE press, 1990).
[2] C.R.Kime, An analysis model for digital system diagnosis, IEEE *Transactions on Computers*, C-19, pp.1063-1073, 1970.
[3] K.R.Pattipati , M.G.Alexandridis, Application of heuristic search and information theory to sequential fault diagnosis, *IEEE Trans. on Systems, Man, and Cybernetics*, Vol. 20, No. 4, pp. 872-887, 1990.
[4] J. D. Brulé, R. A. Johnson, E. J. Kletsky, Diagnosis of equipment failures, *IRE Transaction on Reliability and Quality Control*, Vol. RQC-9, pp. 23-34, 1960.
[5] E. J. Kletsky, An application of the information theory approach to failure diagnosis, *IRE Trans. on Reliability and Quality Control*, Vol. RQC-9, pp. 29-39, 1960.
[6] Johnson R A, *An information theory approach to diagnosis*, Proc. 6th Symposium on Reliability and Quality Control, pp.102-109, 1960.
[7] A. Mahanti, A. Bagchi, AND/OR graph heuristic search methods, *Journal of the ACM*, Vol. 32, No. 1, pp. 28-51, 1985.
[8] A. Biasizzo, A. Zuzek, F. Novak, Sequential Diagnosis with Asymmetrical Tests, *The Computer Journal*, Vol.41, No.3, pp.163-170, 1998.
[9] B. T. Murray, J. P. Hayes, *Test propagation through modules and circuits*, Proceedings of International Test Conference, pp. 748-757, 1991.
[10] C. Robach, P.Malicha, G. Michel, *Computer aided testability and test generation*, Proceedings of International Test Conference, pp.338-345, 1983.
[11] M. Khalil, C.Robach, F.Novak, Diagnosis strategies for hardware or software systems, *Journal of electronic testing: Theory and applications*, Vol. 18, pp. 241-251, 2002.
[12] A. Zuzek, A. Biasizzo, F. Novak, Sequential diagnosis tool, *Microprocessors and Microsystems*, Vol.24, pp.191-197, 2000.
[13] W R Simpson, J W Sheppard, *System Test and Diagnosis* (Kluwer Academic Publishers, Massachusetts, 1994).
[14] J. Park , W. Jung, A study on the systematic framework to develop effective diagnosis procedures of nuclear power plants, *Reliability Engineering & System Safety*, Vol. 84, Issue 3, pp. 319-335, 2004.
[15] O.E. Kundakcioglu, T. Unluyurt, Bottom-Up Construction of Minimum-Cost and/ or Trees for Sequential Fault Diagnosis, *IEEE Trans. on Systems, Man, and Cybernetics*, Part A, Vol. 37, Issue 5, pp. 621-629, 2007.
[16] S. Chessa, P. Santi, Operative diagnosis of graph-based systems with multiple faults, *IEEE Trans. on Systems, Man, and Cybernetics*, Part A, Vol. 31, Issue 2, pp. 112-119, 2001.
[17] Fang Tu, K.R. Pattipati, S. Deb, V.N. Malepati, Computationally efficient algorithms for multiple fault diagnosis in large graph-based systems, *IEEE Trans. on Systems, Man, and Cybernetics*, Part A, Vol. 33, Issue 1, pp.73-85, 2003.

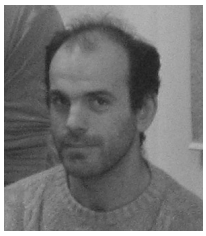
Authors' information

[1]Gorenje d.d., Velenje, Slovenia.
[2]Jozef Stefan Institute, Ljubljana, Slovenia.
[3]Jozef Stefan Institute, Ljubljana, Slovenia.

Peter Mrak is working toward his PhD degree at Jozef Stefan Institute, Ljubljana. He is also with Gorenje d.d. where he is a member of research and development Department of refrigeration appliances. His research interests include electronic test, system diagnosis, control of cooling systems and quality control.



Franc Novak gained BSc, MSc, and PhD degrees in electrical engineering from the University in Ljubljana in 1975, 1977, and 1988, respectively. Since 1975 he has been with the Jožef Stefan Institute, where he is currently head of the Computer Systems Department. Since 2001 he is also an assoc. professor at the Faculty of Electrical Engineering and Computer Science, University of Maribor. His research interests are in the areas of electronic testing and diagnosis, and fault-tolerant computing. His most recent assignment has been on designs for the testability of analogue circuits.



A.Biasizzo has been a researcher at the Jožef Stefan Institute since 1991. He received a Ph.D. degree from the University in Ljubljana in 1998. His research interests include efficient algorithms for sequential diagnosis, constraint logic programming, model-based diagnosis and automatic test-pattern generation.