Modelling and Design of High-Speed Wireline Transceivers with Fully-Adaptive Equalization

Thesis submitted in fulfilment of the requirements for the degree of
Doctor of Philosophy
in Industrial and Information Engineering
at Università degli Studi di Udine

Davide Menin

Udine, 2021

Supervisor
Prof. Dr. Pierpaolo Palestri

Supervisor
Prof. Dr. Pierpaolo Palestri, Università degli Studi di Udine, Italy

Reviewers
Prof. Dr. Salvatore Levantino, Politecnico di Milano, Italy
Prof. Dr. Samuel Palermo, Texas A&M University, USA

Contact
Davide Menin
Università degli Studi di Udine, DPIA
Via delle Scienze 206, Udine (Italy)

Contents

Abstract

High-speed wireline transceivers are analog/mixed-signal electronic circuits in charge of transferring massive amounts of data in a plethora of applications ranging from server racks, base stations of wireless networks, computing platforms and memory interfaces. Recently, automotive applications are emerging as a new field where high-speed data transfer is required to feed information to advanced driver-assistance systems responsible of increasing safety and possibly lead to autonomous vehicles.

In order to design high-speed interfaces complying with automotive standards that demand wide operating conditions and high performance in harsh environments, a full understanding of such devices and of possible impairments is required. This work proposes simple, albeit accurate and computationally-efficient, probabilistic models capable of predicting the deleterious effect of inter-symbol interference caused by the transmissive medium, while replicating the behaviour of the transceiver's circuitry, so to provide a solid background to aid the system/circuit design.

One of the major challenges in designing versatile high-speed interfaces is to cope with inter-symbol interference induced by unknown, high-loss channels. This can be achieved by employing fully-adaptive algorithms to automatically find the optimal settings for the equalizers to compensate the loss and distortion caused by the channel. A method to efficiently simulate such adaptive systems with the aforementioned statistical model is proposed and used to drive the design of an integrated 12 Gb/s PAM-2 transceiver in 28 nm planar CMOS technology, building on a previous chip developed at Infineon Technologies Villach, which was upgraded by modifying design and layout of specific, critical blocks that were identified following its laboratory characterisation. The new test chip was eventually fabricated and characterised to verify the implemented fully-adaptive algorithm when equalizing various channels with different characteristics.

Having guaranteed adaptive equalization, as required in order to comply with modern automotive standards, the 16 Gb/s PAM-2 data rate that they target was considered next. This required to redesign the transmitter from scratch, shifting to a half-rate architecture and implementing circuits to guarantee efficient serialisation, output impedance matching and absence of duty cycle distortion on the 8 GHz clock to achieve the best performance.

The aforementioned work on modelling, design and characterisation of fully-adaptive equalizers for high-speed wireline transceiver was then complemented with modelling activities related to jitter in clock- and data-recovery units, which are responsible for providing a clean and synchronised clock at the receive side to correctly sample the incoming data.

Overall, the PhD work covered many of the most relevant concepts and building blocks of high-speed serial interfaces with modelling, design and laboratory characterisation. This allowed to gain a deep understanding of fundamental aspects such as inter-symbol interference and jitter, how they can be efficiently modelled and how they relate to and interact in the design of a fully-adaptive equalization algorithm and of a 16 Gb/s PAM-2 transmitter in 28 nm planar CMOS technology.

List of Acronyms

ADC	Analog-To-Digital Converter
AFE	Analog Front End
BER	Bit-Error Ratio
CDF	Cumulative Distribution Function
CDR	Clock- and Data-Recovery
CTLE	Continuous-Time Linear Equalizer
DAC	Digital-To-Analog Converter
DCD	Duty Cycle Distortion
DFE	Decision-Feedback Equalizer
FIR	Finite Impulse Response
FFE	Feed-Forward Equalizer
HSIO	High-Speed Input/output
HSSI	High-Speed Serial Interface
ICT	Information and Communication Technology
ISI	Inter-Symbol Interference
LMS	Least-Mean Squares
MSE	Mean Square Error
NRZ	Non-Return to Zero
PAM	Pulse-Amplitude Modulation
PD	Phase Detector
PDF	Probability Density Function
PDN	Power Delivery Network
PI	Phase Interpolator
PLL	Phase-Locked Loop
PRBS	Pseudo-Random Bit Sequence
PSD	Power Spectral Density
PSIJ	Power Supply-Induced Jitter
PVT	Process, Voltage and Temperature
RTL	Register-Transfer Level
SerDes	Serializer-Deserializer
SS-LMS	Sign-Sign Least-Mean Squares
UI	Unit Interval
VGA	Variable-Gain Amplifier
ZF	Zero Forcing

Chapter 1

Introduction to High-Speed Wireline Communications

1.1 Rationale and Historical Trends

The origin of electrical wireline communications can be traced back to the invention of the telegraph in the 1830s, even predating Maxwell's formalisation of electromagnetism, and have evolved ever since to allow transmission of both analog (e.g. the telephone) and digital signals (e.g. the fax machine and, later on, the Digital Subscriber Line) at increasingly higher speeds as novel technologies became available to meet new communications demands [1].

Despite the introduction over time of optical and wireless communication techniques in many fields of *information and communication technology (ICT)*, transmissions over wires remain practically uncontested for many applications where high speed, cost and performance must all be optimised, not to mention the fact that wireline data movement is still required everywhere computation is performed or, in some applications, simply for backward compatibility. In this respect, one of the main driving forces for the improvement of wireline communications in the last decades has been the continuous increase in the number of ICT users and their demand for computational power and data bandwidth, both of them exacerbated by the advent of cloud computing, Internet of Things (IoT) and the related dominant server-centric paradigm of the last decade [1–7]. Obviously, *long-haul* communications are primarily taken care of through wireless and optical means (e.g. mobile and fibre optic networks, satellite links), but local data transfer on the device in use, in base stations for cellular networks, server switches or simply in the interface to optical fibres is mainly operated through wires, chiefly due to the cost (in terms of price, power consumption and chip area) of photonics [2, 7–9]. Such *short-haul* communications targeted by high-speed wireline transceivers are typically intended for *chip-to-chip* and *board-to-board* applications, meaning that they connect different chips that are instantiated on the same board (e.g. processor and memory units), as well as chips that stand on different boards and may be farther apart from each other (e.g. different processing units in vehicles).

By factoring in also the shift towards many-core processor architectures, with their ever higher clock frequency and massive data transfer among different units, one can easily see how the required input/output (I/O) bandwidth has grown so rapidly in order to scale at the same rate [10, 11]. Recently, artificial intelligence and machine learning applications have also started to act as a driving force in this field, due to the large amount of data that they must continuously and efficiently process, transfer and store to complete their training phase [11].

To try and put it in perspective, computer performance (e.g. measured as number of operations per unit time) has kept doubling every 1.5 years for the past 40 years, indicating that system processing performance can be expected to increase by a factor 1000 by 2024 with respect to 2009 [2,11]. Meanwhile, data traffic is expected to increase in the range of several-fold to tenfold every 5 years [2].

In order to follow the aforementioned trends, the data rate per pin has approximately doubled every four years across various I/O standards ranging from double data rate (DDR) to graphics and the high-speed Ethernet [2,7,12]. Trends in specific fields may differ slightly, as is the case in high-performance application-specific integrated circuits (ASICs), where I/O bandwidth is reported to double every two years [5].

However, this race towards faster data transfer faces many challenges and constraints that designers must cope with to provide performing and competitive devices to the market. Below is a short summary of some of the limitations that have to be tackled either at the system or circuit level, and that have concurred in defining and shaping the field of wireline transceivers.

Pin Availability

At a first glance, it seems like that the aforementioned bandwidth demand can be easily met by providing massive I/O parallelism, with each lane supporting low data rates, but a closer look at the matter reveals that the I/O pin availability in modern integrated circuits (ICs) is limited by packaging choices and technologies, leading to double the number of pins per package only every six years, not to mention the need for pins dedicated to other types of I/O and power supplies, resulting in a net increase in bandwidth per pin over time [5]. Other constraints to the available number of I/O pins are the cost of extra wiring or board traces, which may be relevant in the overall system, and the need to save packaging space [13,14].

This has led to widespread adoption of *serial* interfaces to reduce the number of pins required for communication purposes, generally referred to as *high-speed serial interfaces (HSSIs)*, *serializer-deserializer (SerDes)* or *high-speed input/output (HSIO)* to comprise both electrical and optical applications, even though in this thesis only the electrical, *wireline transceiver* case is considered. In the following of this work, these four terms will be used interchangeably.

Constraints on Power Consumption

From a computer-system point of view, the per-computer power consumption remained almost constant for the past 40 years, with only a slight deviation due to a 6 % higher increase rate in power consumption [2]. However, relatively recently (after reaching the 65 nm node) switching energy has started scaling down at a lower rate with decreasing technology nodes (given a scaling factor n, switching energy transitioned from a n^3 to a n decreasing trend), while the ratio between wiring capacitance and gate fan-in capacitance (a metric corresponding to the proportion between switching energy used for on-chip data transfer and computation) increased with the decreasing node due to the larger amount of on-chip interconnections, resulting in energy consumption within processors being dominated by data transfer [2].

The consideration above holds from the standpoint of processors alone, but memory access operations require two to three orders of magnitude higher energy

than compute operations, with data transfer itself accounting for most of the energy consumption [2]. Such chip-to-chip communications are therefore responsible for a large share of the overall system's energy demand and, in general, it can be easily seen that transceiver bandwidth needs not only keep scaling as the computational power and data transfer demands, it has to do so while achieving energy efficiencies (usually measured in mW/(Gb/s) or, equivalently, in pJ/b) that are suitable for integration in such systems, i.e. they must not be the dominant user of energy [8]. Estimates of energy efficiency trends vary, with reported yearly decrease values ranging from 13 % to 30 % [1,2]; however, designers should keep in mind that such a trend needs to compensate the increase in aggregate bandwidth demand, thus taking into account the overall data-transfer power consumption of the entire chip, which in turn is ultimately limited by the thermal characteristics of the packaging technology in use.

Dealing with Channel Loss

Another factor affecting power consumption and the request for energy efficiency is the increased channel loss that transceivers must cope with in order to communicate at higher data rates. In fact, estimates suggest that a 30 dB increase in channel loss leads to a tenfold increment in power consumption [5]; therefore, channel loss is so critical in modern wireline transceivers that a lot of research has focussed on methods to compensate for it [8], and such a topic will be treated in grater detail in the following of this thesis.

1.2 Applications of High-Speed Serial Interfaces

High-speed interfaces are literally ubiquitous in any field of ICT due to the obvious need to transfer information and the ever greater speed at which it is demanded. Therefore, wireline data communications can be found in low-power IoT systems, hand-held electronics as wells as in supercomputers, providing data transfer for applications such as memory, graphics, chip-to-chip fabrics, backplane communications, rack-to-rack connections, Local Area Networks and, emerging recently, automotive applications [7].

Categorising HSSIs usually considers system-level features or the final application itself, providing some insight on requirements, trends and metrics for each category, or it may take into account the length of the physical channel that serial links communicate on, ranging from approximately 1 cm for Ultra-Short Reach (USR) to over 1 m for Long-Reach (LR) links, as a good proxy for typical specifications and requirements [15].

Probably the most generic application field is that of *chip-to-chip communications*, which are key requirements in high-performance computing systems and in modern multi-core processors, or anyway in any system where data transfer among different units at very high data rates is required [10, 11].

Next-generation memory-to-processor standards will require an aggregate bandwidth of 8 Tb/s, while individual line cards in data center switches will scale beyond 51.2 Tb/s data rates [5, 6, 16]. Single links in data center applications are expected to exceed 100 Gb/s and present-day standards support 56 Gb/s communications [4, 6].

An application akin to more conventional chip-to-chip communications is the implementation of *multichip modules* (MCM) as a packaging technique, where limits on die size impose to rethink large systems so to build them as multiple semiconductor dice (possibly fabricated in different technologies) connected using on- and off-package high-speed, short reach links. This allows the various processing units to communicate at data rates close to on-chip total bandwidth and enable massively-scalable parallelism from a design point of view, exceeding data rates of 50 Gb/s per lane with energy efficiencies below 1 pJ/b [11,17].

Quite intuitively, memory applications have always demanded high speed data transfer to and from the processing unit, starting from 400 Mb/s for the first DRAMs in the late 1990s until currently reaching 12 Gb/s, both figures referring to single pins [18]. Peculiar challenges in such applications are the quality of the DRAM technology process for the devices in analog/mixed-signal circuits, which are slower and possibly degraded by the high-temperature fabrication and thus have a hard time working at high speed [18], while signal routing, often performed in multi-drop fashion, heavily impacts the quality of the signal and demand for specific equalization techniques to compensate for this (see Section 1.3.3 for further details) [19].

Graphics is another demanding field for wireline transceivers, as ultra-High Definition (UHD) displays require resolutions of at least 3840x2160 pixels, 10-bit colour depth and 120 Hz frame rate, thus easily exceeding aggregate 10 Gb/s data rates for video data only, i.e. without considering the necessary protocol-defined control and flow bits: 4K UHD televisions typically implement aggregate data rates of approximately 36 Gb/s and are expected to exceed 140 Gb/s to meet the demand for next generation 8K (Quad-UHD) [20,21]. The actual transmission from the controller to the pixel drivers is parallel, but single transceivers composing the bundle need to support beyond-Gb/s speeds and can even exceed 20 Gb/s [20–23]. Moreover, interfaces for display applications must comply with peculiar equalization requirements due to the particular channel architectures [21].

Consumer interfaces are also required to reach very high data rates to support a plethora of everyday applications [24,25]: *Universal Serial Bus* (USB), probably the most famous HSSI, since 2000 has increased its data rate from 480 Mb/s to 40 Gb/s of future 4.0 implementation; one of its main competitor, *Thunderbolt* (TBT), is also widely known and version 3 achieves 20 Gb/s for link reaches below 2 m.

1.2.1 High-Speed Serial Interfaces for Automotive Applications

The automotive industry is a relevant emerging field of application for HSSIS, mainly due to the development and widespread diffusion of infotainment (a portmanteau for *information and entertainment*) and *advanced driver-assistant systems (ADASS)* in recent years, not to mention the huge effort that is currently taking place all over the world in academia as well as in the industry for the development of autonomous vehicles [11,26,27]. These advances are highly motivated by the promise of safer transportation, and currently implemented in the form of systems e.g. for lane-departure and blind-spots warnings, lane-change assistance, adaptive cruise control and pre-collision detection, while looking to the future for systems with the capability to fully control all aspects of driving, from the most *technical* ones (i.e. safely and lawfully reach the chosen destination) to the most *empirical* ones (e.g. human-vehicle behaviour prediction) [26,28].

These advances require to rethink the overall physical and functional electrical architecture of the car: Although primitive computers were on board since the end of the 1960s for tasks such as fuel injection and controlled transmission, modern cars now have to perform massive processing tasks among hundreds ofe-electronic control units (ECUs), while communicating with huge amounts of ADAS sensors, maintaining in-vehicle networks (mainly wired) and connecting with the outside world (i.e. vehicle-to-everything V2X wireless communications, e.g. for over-the-air software updates or fleet management) [14, 27–30]. The car is basically becoming a huge digital system requiring various gigabytes of software (many hundreds of million lines of code) for proper functioning, application-specific processing units and backbone networks capable of guaranteeing high performance and safe, real-time, minimum-latency operations while complying with automotive standards regarding reliability and functional safety [14, 30]; computational power in the order of 10^{12} operations per second (1 TOPS) is needed, hence requiring the transfer of ever larger amount of data[1] at high speed and quality to meet strict constraints regarding functional safety and cybersecurity [14, 27, 30].

The above will be possible thanks to a relevant shift concerning in-vehicle networks, which must still support legacy *signal-based* (often broadcast) communication while implementing new *service-based* IP packet networks. Such advanced automotive protocols are capable of data rates exceeding 10 Gb/s with features that allow priority packets, bandwidth reservation, bounds on latency, precise time synchronization and generally support real-time applications [14, 30].

As of 2016 the most popular in-vehicle networks employed are [14]:

LIN (Local Interconnection Network) for low-speed communication not requiring strict timing performance (e.g. sensors with low refresh rate). It is a low cost solution but limited at data rates of approximately 20 kb/s.

CAN (Controller Area Network) in powertrain and on-board diagnostics. Its low cost, flexibility and easy implementation counterbalance its low bit rate of 500 kb/s and its unsuitability for safety applications.

FlexRay for advanced chassis control and communication with high determinism and fault tolerance at a maximum achievable data rate of 20 Mb/s. It is relatively expensive but has a good flexibility in terms of network topologies.

Ethernet for modern applications' high-speed communication, ECU flashing and data transfer up to 100 Mb/s despite its high but decreasing cost. Protocols extensions that improve fault tolerance for time-critical applications exist.

MOST (Media-Oriented System Transport) for high-bandwidth infotainment and multimedia data. Its 150 Mb/s data rate and good fault tolerance make it suitable for modern applications, although the availability of optical or wireline communication raises its cost.

Other networks such as VAN (Vehicle Area Network), TTCAN (Time-Triggered CAN), CANFD (CAN with Flexible Data rate) and LVDS (Low-Voltage Differential Signalling) are currently in use, but their limited features and inferior performance indicate that they will lose market share in favour of the aforementioned protocols [14]. Nonetheless, as can be seen from Figure 1.1, actual modern in-vehicle

[1] Autonomous cars are expected to individually generate up to 4 TB of data per day [30].

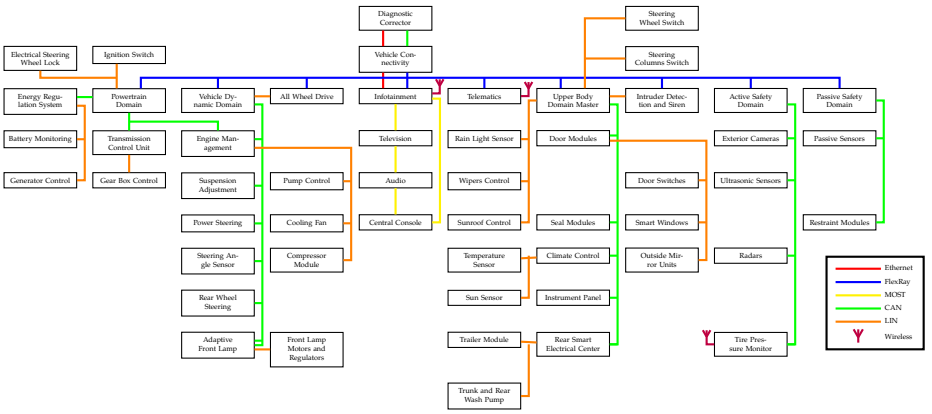


Figure 1.1: Example of in-vehicle network topology. (Adapted from [14].)

networks do not rely on a single solution, but combine many of the implementations above according to systems and subsystems specifications [14].

Note that the discussion above concerns the *protocols* implemented in the networking, which comprise both high-level specifications on packets, collision detection, handshakes, etc. and low-level features such as type of supported transmissive media and various hardware (i.e. electrical) specifications. The latter is what is usually regarded as the *physical* layer (PHY) of the protocol or standard; it is directly implemented in hardware (analog, digital and/or mixed-signal) and it is what this thesis focusses on.

Regardless of popularity or market share, various standards are currently being proposed to overcome the many limitations of the aforementioned protocols and better meet modern vehicles' requirements of multi-Gb/s communication. The most promising ones seem to be Peripheral Component Interconnect Express (PCIe) 4.0, MIPI A-PHY, CoaXPress and Automotive Ethernet, targeting respectively 16 Gb/s, 16 Gb/s, 12.5 Gb/s and 10 Gb/s data rates, while other standards such as HDBaseT, Automotive Pixel Link (APIX), FDP-Link III, FireWire and Gigabit Multimedia Serial Link (GMSL) target relevant, although lower, bitrates of 6 Gb/s, 6 Gb/s, 4 Gb/s, 3.2 Gb/s and 3.12 Gb/s respectively.

1.3 Generic Wireline Communication Systems

Generally speaking, a communication system or transceiver (TRX) is made up of a transmitter (TX), a transmissive media or channel, and a receiver (RX). As schematically depicted in Figure 1.2 for a generic HSSI, such three main blocks may be summarised as [1, 8]:

The transmitter is made up of an encoder for the input data, a serializer to transform input parallel words into a serial stream for transmission, possibly an equalizer to provide pre-emptive correction to the transmitted data, and a driver to impose a voltage or a current to the transmissive medium;

The channel may be a regular copper wire, a PCB microstrip or stripline, or often a combination of all of them when transmit and receive chips stand on different

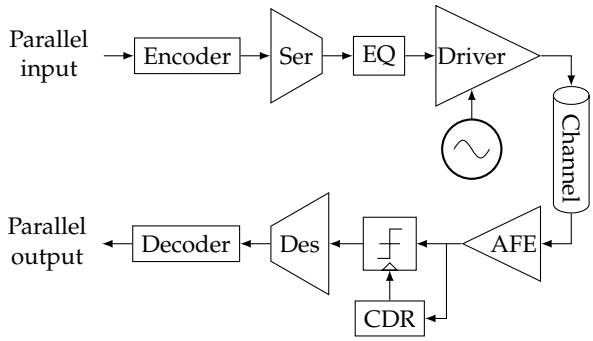


Figure 1.2: Schematic architecture of a high-speed serial interface. Ser and Des stand for *serializer* and *deserializer*, respectively, EQ is the *equalizer*, AFE is the *analog front end* and CDR is the *clock- and data-recovery unit*.

boards, possibly separated by a relevant distance;

The receiver features circuits to condition the incoming signal, equalizers to compensate for losses and distortions induced by the channel, a clock- and data-recovery (CDR) unit to reconstruct the clock and sample the incoming data, a deserializer to turn the input serial data stream into parallel words and finally a decoder to restore the original data format.

A fourth block which is key to ensure good performance in any HSSI application is the *clocking* sub-system, which is divided between transmitter and receiver and determines the timing relationship and synchronization between the two [1, 8].

Yet another "transversal" sub-system is the *equalization* circuitry, which can be implemented both at the transmit and receive side of the HSSI and is in charge of compensating the deleterious effect that the channel has on the communication's quality [8, 31].

1.3.1 Clocking

As briefly mentioned above, the clocking sub-system is of paramount importance for the correct operation of HSSIS. As will be more extensively explained in Sections 1.3.4 and 1.3.5, and in greater detail in Appendix A, it must ensure correct timing and synchronization among all interacting blocks, proper data sampling and retiming in order to allow multi-Gb/s communication. Moreover, it must cope with non-ideal behaviours of the clock waveform that can impair transmission.

1.3.1.1 Jitter

Generally speaking, the concept of *jitter* is used to represent the uncertainty on the edge placement of a clock waveform or, analogously, its deviation from ideal timing [13,32], which may be caused by random or systematic phenomena occurring in the overall circuit (refer to Appendix A.1 for further details). This can be visualised in Figure 1.3, where after a certain clock edge is taken as a reference, as time goes on the timing uncertainty of subsequent edges grows, eventually making it impossible to use such a signal as a timing reference, a synchronising event or as a precise sampling edge. In other words, given a reference initial time t_0 and a real

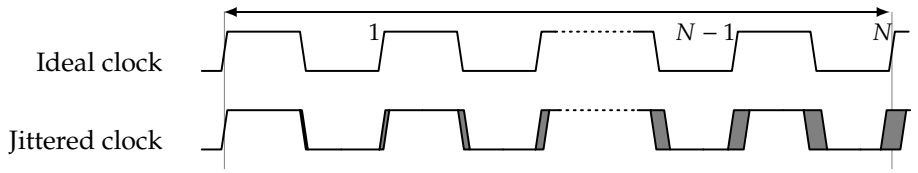


Figure 1.3: Representation of jitter as uncertainty on the clock edge placement.

clock with nominal period T, the actual position of the N-th edge is expected to be in the neighbourhood of the ideal value $t_0 + NT$, but its exact timing cannot be known precisely.

1.3.1.2 Clocking Architectures

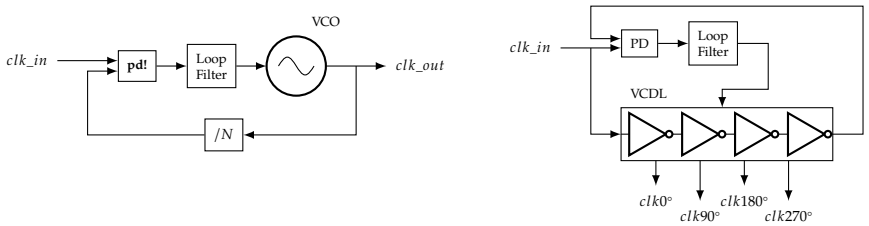
As shown above, clock is subject to non-idealities that affect the whole circuit that is clocked, whose performance is highly dependent on clock quality, so much so that jitter poses upper limits to the data rate that HSSIs can achieve [13]. Providing high-quality clock is thus of paramount importance in all digital applications involving high-speed signalling, from processors to *analog-to-digital converters (ADCs)* and all kind of communication systems, but such a field is broad and it is outside the scope of this thesis to provide a comprehensive treatment of the subject. Therefore, only a brief overview on *clock generation* (or *synthesis*) and *distribution* will be given in this section, while more room is devoted to clock recovery (in Section 1.3.1.3 and especially in Appendix A) due to its importance in SerDes.

Clock synthesis is generally based on oscillators, i.e. circuits with positive feedback that resonate at a fixed frequency, thus providing a periodic clock signal [13]. As mentioned above, fundamental noise sources affecting oscillators are transferred to the clock, thus generating random jitter; if no countermeasures are taken, such a thermal noise-dominated oscillator's jitter accumulates over time and increases in magnitude as the square root of the accumulation time [13].

Such an ever-higher uncertainty regarding the clock edges can be unbearable in many applications, demanding more sophisticated solutions to keep jitter below an acceptable level. Often, oscillators are used to provide a reference clock to control loops that are capable of tracking both phase and frequency to achieve the desired performance, a circuit called *phase-locked loop (PLL)* shown in Figure 1.4a [13]. Such circuits attenuate high-frequency jitter up to their loop bandwidth while tracking low-frequency components as well; moreover, they are particularly useful for frequency multiplication starting from a low noise, lower frequency reference [13].

When multiphase clocks are required, two solutions are commonly available: *Delay-locked loops* (DLL) are based on a controlled delay line which can be tapped at different nodes to output different phases of the same clock, as in Figure 1.4b; while *phase interpolators (PIs)* produce a specified weighted mix of two clock phases, thus generating any possible phase between the original two [13].

Clock distribution in the chip is another topic of particular concern, because it is common practice to instantiate a single clock synthesiser (or just a few) in order to reduce power and area costs and then route the clock to multiple circuits that need it [13,33]. Such distribution networks are made up of metal interconnections and active buffers: The former are bandwidth-limited channels akin to those of Section 1.3.2, which cause jitter amplification especially with high-frequency



(a) Phase-Locked Loop (b) Delay-Locked Loop

Figure 1.4: Block diagram of simplified (a) PLL and (b) DLL; PD is the *phase detector*, VCO is the voltage-controlled oscillator, $/N$ is the frequency divider and VCDL is the voltage-controlled delay line.

clock; the latter are subject to *power supply-induced jitter (PSIJ)* due to the chip's own activity, which is translated from supply noise to clock timing uncertainty [13, 33] (refer to Appendix A.4 for further details).

Careful design and layout of clock distribution networks are therefore necessary to mitigate the above, and often consist of difficult trade-offs in power, area and cost optimisation [13, 33]. Advanced solutions include implementing dedicated supply regulators for the distribution circuitry, adding PLLs or injection-locked oscillators in the network in order to produce a low-pass filter for the clock's jitter or designing resonant clocking and distributed oscillators [13, 33].

1.3.1.3 Introduction to Clock and Data Recovery

Clock- and data-recovery (CDR) circuits are of paramount importance in HSSIS: They take care of providing a clock at the receiver that is synchronous to the incoming data so to sample the received bit stream without errors. Such a clock cannot simply be generated locally and used as is because mismatches in the clock synthesis circuits at transmitter and receiver make it impossible to obtain two waveforms oscillating at precisely the same frequency, therefore the sampling clock would periodically miss a bit and increase the error count. As an example, a 100 ppm (*parts per million*) oscillator slips one *unit interval (UI)* (corresponding to the width $T_B = 1/f_B$ of transmitted bits, where f_B is the data rate) every 10 000 bits, which translates to one error every µs if transmitting at 10 Gb/s [34], i.e. one million errors per second! Moreover, a clock source may not even be available at the receiver, so that it has to be extracted from information sent by the transmitter [34].

In order to reduce the amount of errors caused by sampling, the receiver clock has to be continuously synchronised to the incoming bit sequence so to guarantee that the sampling point does not drift with time and that it is kept at the centre of the received bit. The most straightforward way to accomplish this is to simply transmit the very clock that was used at the transmitter, since it is synchronous to the data stream by construction. This is done in *forwarded-clock* (or *source-synchronous*) architectures shown in Figure 1.5a, which employ a separate pin and channel for clock transmission and can ensure jitter tracking between data and receiver sampling clock, if the two paths are well matched [13]. However, pin availability may not allow the use of a dedicated channel for clock transmission, especially if such a cost (and that of the increased power consumed to transmit the

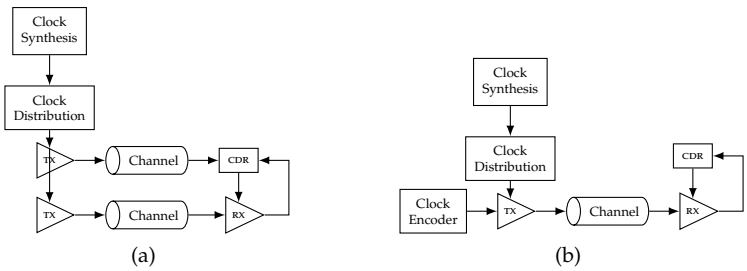


Figure 1.5: Two of the most common clocking approaches in wireline transceivers: (a) Forwarded- and (b) embedded-clock architectures.

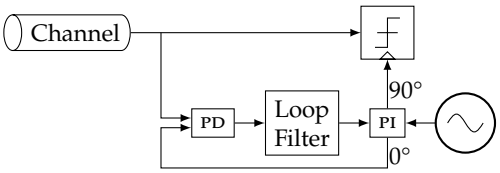


Figure 1.6: Block diagram of a simplified CDR, showing the operating principle of clock recovery. PD is the *phase detector*, PI is the *phase interpolator*.

clock) cannot be amortized among many transceiver lanes.

Embedded-clock (or *plesyochronous*) architectures avoid the need for a dedicated clock channel by reconstructing the clock from the received bit stream and then using it for sampling (Figure 1.5b). Such a solution requires a clock source and a method to continuously synchronise it to the incoming data sequence, so that frequency differences between transmitter and receiver clocks do not affect sampling. This takes the form of a *clock- and data-recovery* unit (CDR, which will be thoroughly discussed in Appendix A), a system akin to a phase-locked loop in structure and principle of operation. As schematically shown in Figure 1.6 for a simplified PI-based CDR, the clock generated locally is compared to the incoming data signal and, after filtering the result of such a comparison, a control signal sets the correct clock phase for optimum sampling at the receiver's slicers [35]. In this way, the signal's zero crossings are exploited to determine the timing information on the embedded clock, and the proper sampling instant is then imposed midway between such transitions.

1.3.2 Channel

As briefly mentioned above, the transmissive medium or channel may take on various forms, from common copper wires to PCB traces, more often a combination of many different media including their connectors: Bumps, solder balls, vias, chip pads, etc. [6, 8, 36]. Generally speaking, whatever the actual channel is, the operating frequencies of typical HSSIS are so high that the wave nature of the signal must be considered, although it is not necessary to deal with the full electromagnetic problem, so that the link is usually modelled as a transmission line with lumped, frequency-dependent per-unit-length $rcgl$ elements [8], as depicted in Figure 1.7.

In order to determine useful relationships for such $rcgl$ parameters, some phys-

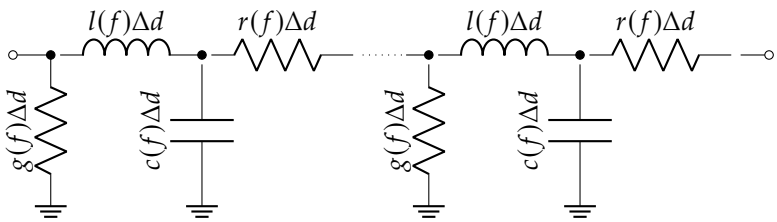


Figure 1.7: Lumped-element approximation for a generic transmission line; $rcgl$ are the per-unit-length parameters, Δd is the length of the line element considered.

ical insight into the mechanism of electromagnetic wave propagation on transmission lines is required, in particular concerning the phenomena of *skin effect* and *dielectric losses*. Skin effect is a direct consequence of the high frequencies involved, which constrain the current to flow through the periphery of the conductors' cross section (indeed, its "skin"), hence reducing its available area and causing the associated resistance to increase as a function of frequency [8,37]. Moreover, since the conductor's interior is devoid of flowing current, its effective inductance decreases due to the absence of induced magnetic field. Such effects are usually modelled as

$$r(f) = r_0 + \sqrt{f}(1+i)r_S, \tag{1.1}$$

where r_0 is the per-unit-length DC resistance and r_S is a skin effect constant. Note that the $\sqrt{f}$ dependency implies a loss that increases at 10 dB/dec [8].

Dielectric losses are then taken into account by exploiting the concept of complex permittivity, which states that at high frequencies

$$\varepsilon(f) = \varepsilon'(f) - i\varepsilon''(f). \tag{1.2}$$

As a consequence, a current $i\omega c(\omega)V(\omega)$ across the capacitance (which depends on $\varepsilon'[\omega]$) is present along with a current $G(\omega)V(\omega)$, with $G(\omega) = [\varepsilon''(\omega)/\varepsilon'(\omega)]\omega c(\omega)$, across the conductance [38].

As a consequence of this lumped-element modelling approach, one finds that signals travelling along the channel may be reflected at either ends, so that care must be taken in order to prevent it from impairing the signal's integrity. Hence, HSSIs are usually designed for a specific characteristic impedance Z_0 (usually 50 Ω) at the frequency of interest and properly terminated at both ends to minimise reflections [8,37,39], as shown in Figure 1.8; however, such termination resistances have no effect on the aforementioned impedance discontinuities caused by connector elements. Such considerations have a direct impact on the link's power consumption: Given a fixed characteristic impedance, the power consumed in the termination impedances V^2/Z_0 can only be minimised by reducing the signal's voltage swing, although at the cost of decreasing the signal-to-noise ratio (SNR); when $Z_0 = 50\,\Omega$, a decent trade off is usually achieved by employing signal swings of a few hundreds of millivolts [8].

In addition to such non-idealities inherent in the physics of electromagnetic waves propagation, application-specific impairments are caused by the presence of various connectors linking the different transmissive media in any HSSI. As briefly mentioned above, and schematically shown in Figure 1.9, the signal path usually crosses many packages, chip pads, solder bumps, sockets, vias, etc., all of

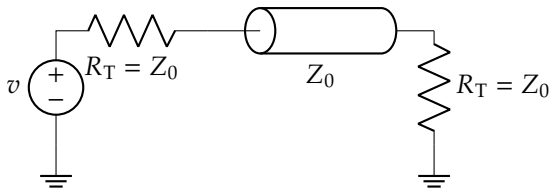


Figure 1.8: When transmitting at frequencies whose wavelengths are comparable with the channel length, the link must be properly terminated to avoid signal reflections.

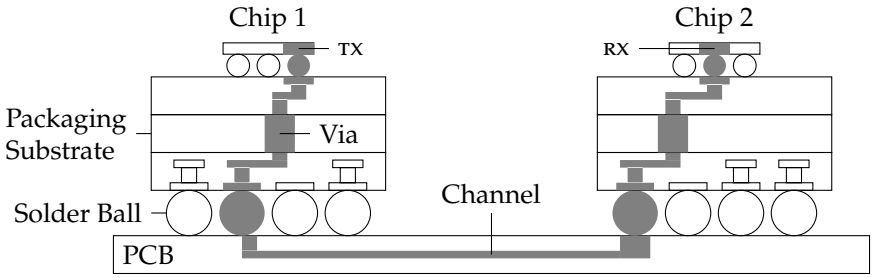


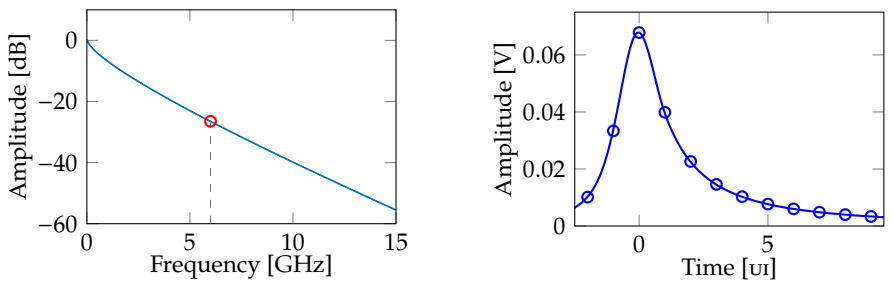
Figure 1.9: Typical chip-to-chip system showing the signal's path through various channels and connectors sharing the same PCB [5, 36]. (Drawing not on scale.)

which introduce discontinuities in the overall channel's characteristic impedance and produce signal reflections that impair the signal's integrity and, in the worst case scenario, may even give rise to standing waves that attenuate portions of the spectrum [8, 36, 37]. Such issues are exacerbated by imperfect transmitter's output return loss, which does not completely absorb the reflected signals and allows them to bounce back towards the receiver, and by the channel characteristics: Low-loss links allow such reflections to travel unhampered, while high-loss ones make them appear negligible when they reach the receiver [36, 37].

1.3.2.1 Inter-Symbol Interference

The lumped-elements model of Figure 1.7 and frequency dependency of the $rcgl$ elements entail an overall low-pass behaviour of the channel's frequency response, as can be seen in Figure 1.10, which in turn limits the bandwidth available to the HSSI. From a frequency-domain point of view, the signal's various frequency components are subject to different attenuations, which means that data sequences with many *consecutive identical binary digits (CIDS)* are attenuated less than sequences with frequently toggling bits; the maximum attenuation affects only the clock-like sequence (...010101...), which excites the channel at the maximum achievable frequency corresponding to half the maximum data rate, usually referred to as the *Nyquist frequency* and denoted as $f_{\text{Nyq}} \triangleq f_B/2$ [8]. For this reason, channel loss is usually reported by stating its loss (in dB) "at the Nyquist frequency" (or simply "at Nyquist").

Not only channels attenuate the transmitted signals non-uniformly over frequency, their non-linear phase response $\Phi(\omega)$ characteristics further result in signal distortion because phase delay $P(\omega) \triangleq -\frac{\Phi(\omega)}{\omega}$ and group delay $G(\omega) \triangleq -\frac{\partial\Phi(\omega)}{\partial\omega}$ are



(a) (b)

Figure 1.10: 12 Gb/s transmission: (a) attenuation vs. frequency and (b) pulse response for a template channel losing −26.8 dB at the Nyquist frequency.

not constant with frequency and give rise to dispersion in the travelling wave [40].

From a time-domain standpoint, the aforementioned frequency-dependent attenuation smoothens the signal's rising and falling edges, and spreads the energy of the transmitted pulse on its sides, as can be seen from the *pulse response* (also called *single-bit response*, SBR) in Figure 1.10, which corresponds to the action of the channel on a trapezoidal pulse representing a realistic transmitted waveform.[2] The peak of the pulse response corresponds to the value that is ideally sampled by the receiver if the CDR can align the receiver clock properly, and it is referred to as the *main cursor*, while other non-zero samples located at multiples of a *unit interval* (UI, equal to $1/f_B$) are called *pre-cursors* if they precede the main cursor (i.e. they are received earlier on), *post-cursors* otherwise [8].

Such a spread of the transmitted pulse, represented by pre- and post-cursors, causes single neighbouring bits to overlap at the receiver, thus altering the values of the sampled voltage, a phenomenon known as *inter-symbol interference (ISI)*. As a consequence, neighbouring bits may destructively interfere one with the other and modify the amplitude of any signal bit so much that they overcome its main cursor, making it appear as of a different symbol and causing a reception error to occur, as shown in Figure 1.11. Therefore, ISI is (together with jitter) the main responsible for communication errors and limits the achievable data rate of any wireline transceiver to a few Gb/s, if action is not taken to mitigate it [8, 31].

Impedance discontinuities and the reflections they cause also contribute to ISI, their impact being highly dependent on the channel features and characteristics of the discontinuities themselves, as mentioned above. Moreover, the arrival time at the receiver of such reflected signals (with respect to that of the main cursor) depends on the channel length and on the location of the discontinuity along it, and may be estimated at first order as

$$T_r = \frac{2d_{\text{disc}}}{c}, \tag{1.3}$$

[2]Note that this definition is only conceptually similar to the *impulse response* commonly used in control and system theory, where one form of time-domain response of linear, time-invariant systems is obtained by applying a Dirac's delta function at the input. In the context of HSSIS, the use of the pulse response is justified by its more direct connection to the transmission of practical data signals, which provides more useful insight on the channel's behaviour. Anyway, such a pulse response is simply the convolution of a the channel's impulse response and the transmitted pulse.

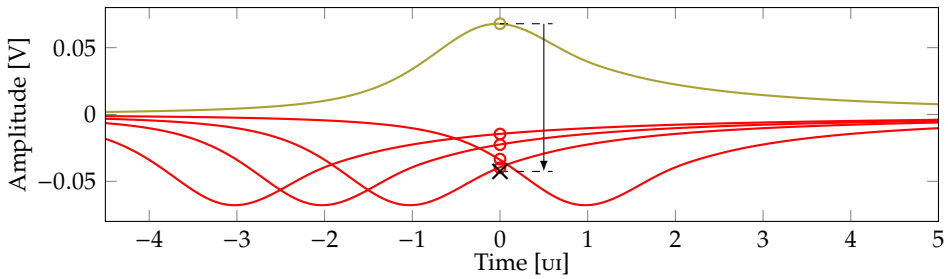


Figure 1.11: Example of error at the receiver caused by ISI when transmitting a '00010' sequence at 12 Gb/s over the channel of Figure 1.10: The main cursor of the '1' bit (green mark) is degraded by the negative contributions of the neighbouring '0' bits (red marks) and results inverted in sign (black cross), yielding an error when sampled. The example shows a case of PAM-2 (*pulse-amplitude modulation*) signalling, where '1' and '0' bits are mapped to two different values, respectively '+1' and '-1', in order to result in two distinct voltage levels at the driver's output.

where d_{disc} is the distance of the discontinuity from the transmitter and c is the wave velocity. This entails that each discontinuity causes reflections that arrive with different delays at the receiver, and such reflections are the farther from the main cursor the longer the channel is; therefore, they will affect faraway cursors (e.g. 12.6 UI for a 25 mm chip-to-chip link transmitting at 28 Gb/s [36]) so that counteracting their degrading effect is rather challenging [36, 41].

1.3.2.2 Introduction to Equalization

Since ISI is a major impairment to achieving high-speed communications, compensating its effects is key in modern transceivers and such a task is addressed through *equalizers* aimed at restoring lost components of the signal spectrum [8]. However, before delving into the problem of equalization, the designer needs a clear understanding of how to determine the effect of ISI and whether it has been compensated or not.

From a frequency-domain point of view, one may intuitively observe the overall frequency response of the system composed of transceiver (comprised of equalizers) and channel: If the combination of the two yields, ideally, a flat frequency response up to much beyond the Nyquist frequency for the case of interest, then the most relevant attenuation and distortion may be expected to be eliminated by the equalizer, as shown in Figure 1.12a [8, 31].

In time domain, as can be seen from Figure 1.12b, one may simply observe the pulse response and require that all pre- and post-cursors be null, or at least approximately so, due to the action of the equalizers, thus entailing absence of ISI [8, 31]. This corresponds to the well-known concept of *Nyquist pulse* [42, 43].

Both the aforementioned methods are very profitable from a theoretical and system-level standpoint, but they are quite unhandy from the point of view of circuit design, simulation and laboratory practice (even though they can be rather useful also in such contexts). The simplest method to determine whether the HSSI under test is working properly or not is probably to just count the number of wrong bits that are received, which means that ISI has caused an error to occur, assuming the clock to be correctly locked to the received waveform. Such a performance is

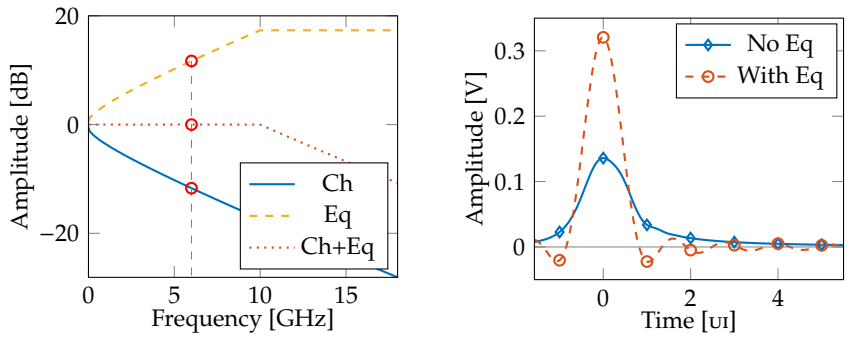


Figure 1.12: Equalization from the point of view of the channel's (a) transfer function and (b) pulse response when transmitting at 12 Gb/s (Nyquist frequency of 6 GHz). With equalization, the combined frequency response is flat up to approximately 10 GHz, causing most of the pulse response's ISI cursors to be closer to zero, while the main cursor more than doubled in amplitude.

measured by the *bit-error ratio (BER)*, defined as the ratio between the number of wrong bits and the total number of received bits.[3] Modern standards often require HSSIS to guarantee BERS below 10^{-12} or even 10^{-15}.

Despite being an important metric in evaluating the quality of communication in SerDes, BER is poorly informative on how the transceiver is actually performing, i.e. it does not provide much knowledge regarding what is affecting the transmission and what can be done about it. An important tool in this respect is the *eye diagram*, which is obtained by folding the received analog waveform onto itself in time slots spanning 1 UI (shown in Figure 1.13), thus allowing the observation of all data transitions that take place in the transceiver in order to assess the effect of the channel [31, 44]. From this plot, relevant figures of merit such as eye height and width can be reported to characterise the eye aperture and indicate available margins for correct detection of the signal's polarity and for correct sampling, respectively. Qualitative information regarding the received signal's slope, the effect of equalization and the amount of incoming signal's jitter (due to the transmitter's clock) may also be obtained from the eye diagram.

The *bathtub plot*, which has BER on its ordinate and sampling position on its abscissa, provides valuable information on the effect of sampling clock jitter on the receiver's operations [31]. As can be seen in Figure 1.14, it can be thought of as a 0 V cross section of the eye diagram, representative of the situation experienced at the receiver's threshold for data slicing in a *pulse-amplitude modulation (PAM)*-2 transceiver (see Section 1.3.5). By observing the behaviour of the BER when moving away from the centre of the plot (which corresponds to the CDR's ideal lock point or the centre of the eye diagram) and towards the data transitions, an analysis of the jitter sources affecting the sampling clock can be carried out (as will be done in

[3] In the literature this is also commonly referred to as bit-error *rate*; however, the word "rate" would entail its definition to be the number of received wrong bits per unit time, hence with unit s^{-1}, but this is seldom the case. Considering that the two definitions are related with each other as BER=(bit-error rate)/(data rate) and that BER is properly independent of the data rate, this thesis will stick to the definition given in the text.

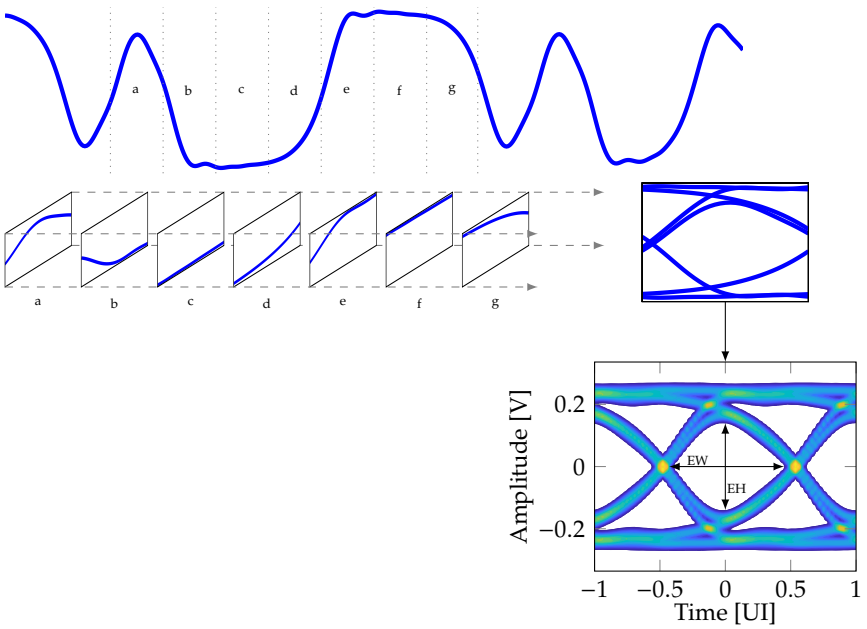


Figure 1.13: Construction of the eye diagram and relevant figures of merit employed to characterise the effect of equalization and transmitter's clock jitter: EH is the eye height, EW is the eye width. The different eye slices are determined by a circuit (the clock- and data-recovery unit) that tracks signal transitions to determine the sampling point.

Section 3.3.2) in order to determine their significance and possible minimisation. Moreover, the effect of equalization techniques can be evaluated by observing the variation in bathtub's width at specified values of BER.

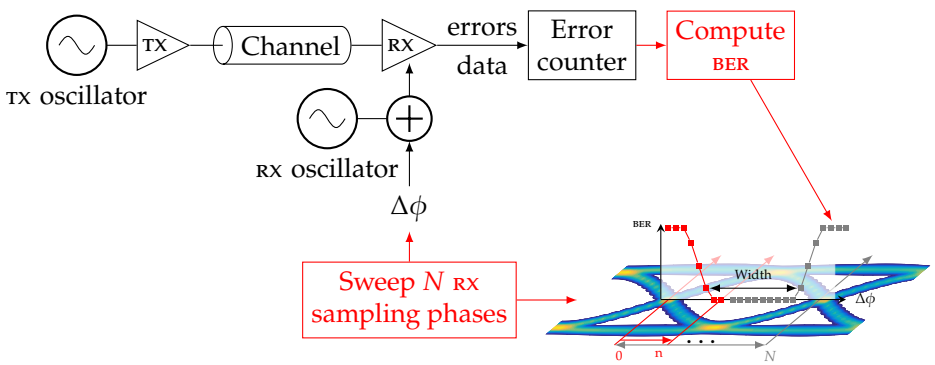


Figure 1.14: Construction of the bathtub plot and relevant figures of merit employed to characterise the effect of equalization and sampling clock jitter: At the receiver, the sampling phase is swept along one UI, the resulting data and errors that are received are accumulated and employed to compute the BER, which is eventually plotted as a function of the sampling phase.

Any of the above can be considered when designing equalizers to combat ISI, since they all provide some insight into the effect of the channel on transmission and of the equalizers in counteracting it. However, as will be seen in Section 1.3.3, each tool has pros and cons that make them more suitable for evaluating certain equalization techniques than others.

1.3.3 Equalization Techniques

A general, system-level treatment of the most common equalization solutions employed in HSSIs is presented in this section, while circuit-level schematics of some equalizers are provided in Sections 4.2.1, 4.2.2 and 6.1, where the blocks in which they are actually embedded are described.

1.3.3.1 Continuous-Time Linear Equalization (CTLE)

Conceptually speaking, *continuous-time linear equalizers (CTLEs)* are rather simple [31]: Since the channel can be modelled as a low-pass filter (as explained in Section 1.3.2) with transfer function $H_{\text{ch}}(f)$, its effect can be ideally compensated for with an analog filter having transfer function $H_{\text{ch}}^{-1}(f)$, for the combination of the two yields unity gain at all frequencies, as in Figure 1.15a. However, any real circuit is limited in bandwidth, thus degrading the compensating effect at high frequency, as shown in Figure 1.15b. Fortunately, perfect compensation is often unnecessary and a flat frequency response up to much beyond the Nyquist frequency suffices for near-optimal results, while effective equalization (open eye diagram with acceptable rise and fall times) can be obtained in practice by compensating the channel loss up to approximately 70 % of the data rate, i.e. $1.4 f_{\text{Nyq}}$ [8, 31].

Practical CTLEs are usually described in terms of their poles and zeros and in their simplest form they are implemented as high-pass RC filters, although (as can be seen in Figure 1.16a) such passive equalizers provide only *de-emphasis*, i.e. they attenuate low-frequency components while leaving the high frequencies untouched [31]. When the target channel loss exceeds approximately 20 dB at the Nyquist frequency, CTLEs capable of providing high-frequency amplification are re-

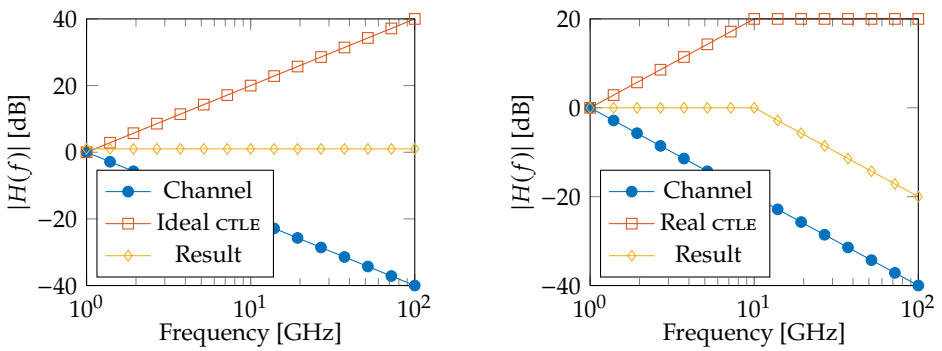


Figure 1.15: Linear equalization shown through the operation of (a) an ideal and (b) a bandwidth-limited CTLE for channel loss compensation.

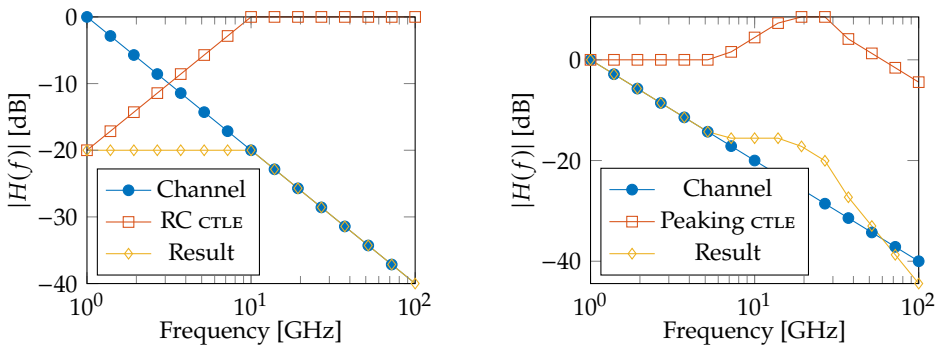


Figure 1.16: The different effect of (a) a simple passive CTLE and (b) one implementing high-frequency peaking.

quired in order to improve the receiver's sensitivity (see Figure 1.16b), which would be hampered by the aforementioned low-frequency attenuation [31]. In such a case, active circuits called *peaking amplifiers* are implemented, whose high-frequency boost performance can be enhanced by resorting to cascaded and/or feedback solutions, which become necessary as the required peaking exceeds 4–6 dB [31].

The choice of a CTLE's parameters is made based on the target channel to be equalized, chiefly aiming at flattening the transfer function at least up to $1.4 f_{\mathrm{Nyq}}$, as mentioned above. The amount of DC gain to implement is typically variable as well, and can be chosen to better accomodate the signal swing for subsequent stages.

1.3.3.2 Feed-Forward Equalization (FFE)

With lossy channels, a given data bit is affected by any transmitted bit preceding or following it (at least by a certain number of them, determined by the quantity of non-zero cursors in the channel's pulse response), so that their effect can be counterbalanced by applying corrections according to their sequence. In practice, this is done by implementing *transversal filters* as that of Figure 1.17, in which the input sequence of the sampled signal x_i is fed through various delay stages before being combined with weighted *tap* coefficients w_i; the largest coefficient is the *main tap* and corresponds to the main cursor of the pulse response, while those with longer and shorter delays are called *pre-* and *post-taps*, respectively, whose value is chosen to compensate the post- and pre-cursors ISI [31].

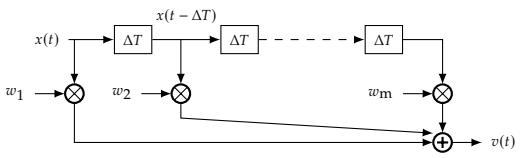


Figure 1.17: Schematic representation of a feed-forward equalizer; the delay ΔT is equal to 1 UI.

Due to their structure, *feed-forward equalizers (FFEs)* are *finite impulse response (FIR)* filters having duration $(m-1)\Delta T$, where ΔT is the delay of each stage, usually equal to 1 UI to reduce circuit complexity, although finer spacing would allow better tuning of the equalizer to the channel [31]. Such a filter can be applied at either ends of the channel due to its linearity, but transmit-side FFE is far more common due to its simplicity: At the transmitter, the signal to be processed is a single digital bit ($x_i = d_i$, the bit values to be transmitted) so that the delay stages are simple latches or flip-flops, tap multiplication is performed on binary values and the final summation is usually performed by simply connecting together the tap drivers' outputs [31]. Instead, the receive-side FFE has to operate on analog signals, hence employing analog delays and analog tap multiplication, all rather challenging from a designer's perspective [31]; however, the recent trend towards analog-to-digital converter-based (ADC-based) HSSIS simplifies such an architectural choice by implementing receive-side FFE through digital adders and multipliers following an ADC, as will be briefly explained in Section 1.4.

One major drawback of all linear equalizers (FFE and CTLE alike) is that they cannot discriminate between signal and noise: CTLE's high-frequency boost compensates the channel loss but amplifies high-frequency noise such as crosstalk or transistor noise, so that for cases with high channel loss the SNR would be drastically reduced; receive-side FFEs also suffer from such noise amplification since they operate on analog signals, while transmit-side FFEs process digital signals and are therefore less susceptible to noise [31].

Another severe limitation of FFE is consequence of the constraint on the output driver's peak amplitude, which entails that the sum of all taps must be constant at all times [45]: When the available power is distributed across different taps in order to cancel various ISI cursors, this is done at the expense of the main cursor's amplitude, which is reduced and implies a lower transmitted signal swing [46]. In fact, this greatly limits the amount of ISI that can be compensated by the FFE when dealing with high-loss channels, and calls for implementing different equalization techniques to aid its task.

The effect of the FFE is commonly referred to as *pre-emphasis* or *de-emphasis*, respectively when only the first pre-tap or only the first post-tap is enabled. These can be quantified by a measure of the attenuation produced by the FFE correction (given that only two taps are employed):

$$A_{\text{pre-emph}} = -20\log\left(\frac{|w_0| - |w_{-1}|}{|w_0| + |w_{-1}|}\right), \tag{1.4a}$$

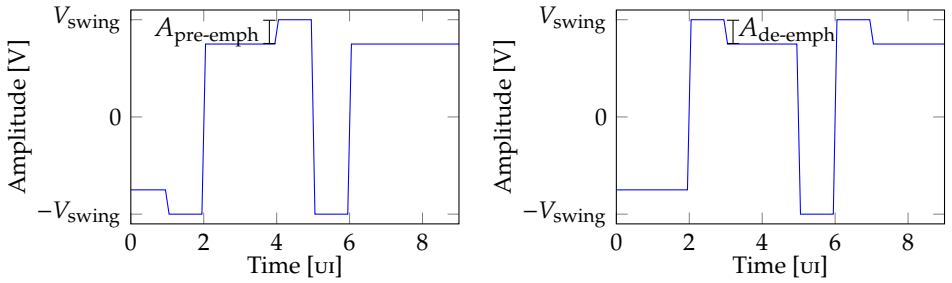
$$A_{\text{de-emph}} = -20\log\left(\frac{|w_0| - |w_1|}{|w_0| + |w_1|}\right), \tag{1.4b}$$

where $|w_0| + |w_{-1/1}| = 1$, but was made explicit for the sake of completeness. An example is shown in Figure 1.18, where the aforementioned swing reduction due to FFE is also visible.

Referring to Figure 1.17, the output of a generic FFE is simply given by the convolution between the input data sequence and the FIR coefficients:

$$v(iT_{\text{B}}) = \sum_k x_k w_{i-k}. \tag{1.5}$$

In order for the FFE to equalize the channel, all relevant K ISI cursors have to be modified to obtain a Nyquist pulse, but an M-tap FFE produces an output sequence



(a) Pre-Emphasis (b) De-Emphasis

Figure 1.18: Examples of FFE pre- and de-emphasis with $w_0 = 0.875$ and $w_{-1/1} = -0.125$, translating to $A_{\text{pre-emph}} = A_{\text{de-emph}} \approx -2.5\,\text{dB}$.

of length $N = K + M - 1$ generally larger than M, thus yielding a system of N equations in M unknowns when trying to *zero-force* all pre- and post-cursors. This not only entails that not all ISI can be equalized (so called *residual ISI*), but also that it is reshaped and some cursors outside the reach of the FFE become more relevant [43].

This is the reason why the most common method of tap weight choice is the *mean square error* (MSE), which consists of minimising the difference between the channel equalized with FFE and an ideal Nyquist pulse z_{Nyq}:

$$\min_{w} \|z_{\text{Nyq}} - \mathbf{H}_{\text{ch}} w\|^2, \tag{1.6}$$

where $\mathbf{H}_{\text{ch}}$ is the convolution matrix for the channel's pulse response. It can be shown that Equation 1.6 is minimised by [43, 47]

$$w_{\text{MSE}} = (H_{\text{ch}}^T H_{\text{ch}})^{-1} H_{\text{ch}}^T z_{\text{Nyq}}. \tag{1.7}$$

1.3.3.3 Decision-Feedback Equalization (DFE)

The *decision-feedback equalizer (DFE)* is based on the idea that the ISI induced by the bits preceding a certain data bit detected at the receiver (i.e. when a decision is taken on the corresponding symbol) can be determined from the channel pulse response [31]. In other words, from the channel SBR the effect that any received bit will have on the following data bits can be determined and compensated. In practice, structures similar to those of Figure 1.19 are implemented, thanks to which post-cursors ISI can be cancelled by storing the sequences of past bits (hence, they are delayed by $\Delta T = 1\,\text{UI}$), weighting them with tap coefficients and feeding them back in order to subtract them from the input analog data before a decision on the incoming bit is taken [31].

The main feature of the DFE is that it is a non-linear equalizer since it works on hard data decisions (i.e. binary data), entailing absence of noise in the feedback data and no noise amplification in its operation [31]. Moreover, it is also very effective in dealing with reflections caused by impedance discontinuities, if the corresponding ISI is comprised in the time span covered by the DFE [31].

A consequence of its working principle is that DFEs cannot compensate for pre-cursors ISI, since such equalization schemes only affect future bits; therefore,

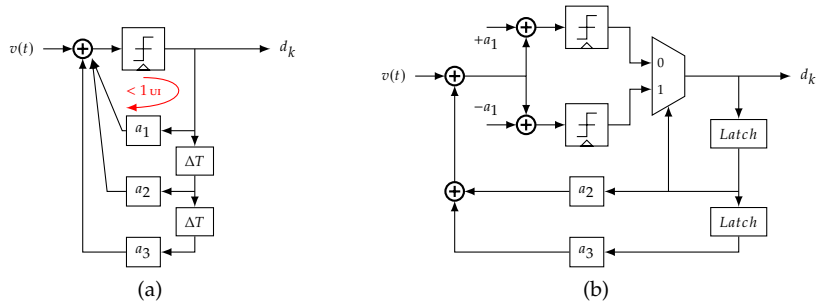


Figure 1.19: Schematic representation of a decision-feedback equalizer (a) and a version with one speculative tap (b) to alleviate the timing issue indicated by the red arrow in the left plot.

they are usually complemented with linear equalizers [31]. Moreover, DFEs are susceptible to error propagation due to their architecture, because a single mistaken decision will be fed back and corrupt the following compensations on the incoming signal for the whole length of the DFE shift register, thus generating error bursts; this might increase BER but can be mitigated by employing forward error correction codes in the communication [31].

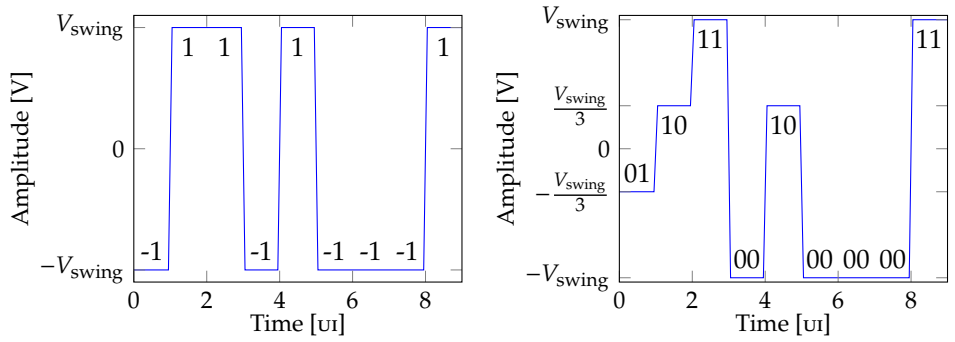
The major challenge in designing a DFE is the tight timing margin available for the feedback of the first tap to settle at the summer's output, since such an operation has to be completed in less than 1 UI [31], as shown in Figure 1.19a. A widespread technique to relax the timing constraint on the first tap is *speculation* (also called *loop-unrolling*), which is shown in Figure 1.19b and implements two parallel summer stages: One applies the first tap assuming that the previous received bit were a '1', the other that it were a '0'; since all received bits are binary data, one of the two represents the correct polarity of first-tap compensation [31]. In this manner, at the following clock edge two decisions are simultaneously taken, corresponding to both possibilities, and the correct one is selected through a MUX once the previous data bit is actually known. Speculation obviously shifts the critical path to the second tap feedback loop, but such a technique can be extended to cover any number of DFE taps, although the number of parallel speculative paths (hence, slicers) grows as 2^S and that of summers as $\sum_i^S 2^i$, with S number of speculative taps, entailing an increase in area occupation and power consumption [6, 31].

The choice of DFE taps is rather straightforward: As they directly compensate the effect of ISI post cursors, their value is set to equal such voltages.

1.3.4 Transmitter

The transmitter's role is, simply speaking, to receive an input digital word and send it as a serial stream according to a defined symbol alphabet composed of possibly various high- and low-voltage levels. The signalling method is defined by the reference standard, with the most common solutions being *differential* and *single-ended*, employing respectively a pair of coupled wires on which complementary data are sent on each one and a single, usually ground-referenced, channel [5, 8].

Differential signalling is the most common choice in all applications where tolerance to induced noise and/or crosstalk is of paramount importance, while



(a) PAM-2 (b) PAM-4

Figure 1.20: Differential pulse amplitude modulation on 2 (often referred to as non-return to zero) and 4 levels, with the latter doubling the amount of transmitted data per UI.

single-ended signalling allows a reduced number of pins and wires (hence, higher bandwidth density, measured in (b/s)/mm^2 of chip area and/or (b/s)/mm of chip periphery) and, in limited usage scenarios such as ultra-short reach links with strict constraints on pin density, may even provide better noise immunity compared to differential signalling [5,17].

Standards define symbol encoding as well, the most common of which is the straightforward PAM-2 or *non-return to zero (NRZ)*[4] where a single bit is mapped during a UI to one of two available voltage levels, usually denoted either as '+1' and '-1' (as in Figure 1.20a) or '1' and '0'. This allows for simple circuit solutions as all that is needed is digital logics to process such binary digits but, as data rates increase, channel loss at the Nyquist frequency grows higher and poses challenges in achieving the target BER [8]. Therefore, modern standards introduce PAM-4 (*pulse amplitude modulation* on 4 levels, shown in Figure 1.20b) that maps two bits onto four-level symbols, thus decreasing by a factor of 2 the Nyquist frequency for a certain data rate and reducing the channel loss affecting the transmission. Despite this clear advantage in terms of equalization requirements and communication quality, PAM-4 transmitters face linearity challenges to generate equidistant voltage levels and more complex architectures to process multi-bit symbols; moreover, reduced distance to the different voltage levels makes PAM-4 signalling more sensitive to noise and ISI [11,46].

The overall transmitter may be regarded as *full rate* (Figure 1.21a) when its components are clocked at a frequency corresponding to the actual data rate of the HSSI using rising (or falling) edges only, this being the simplest solution from an architectural point of view, despite the challenges of designing analog circuits working at such high frequencies and respecting all timing margins; *half rate* (Figure 1.21b) when a clock divided by two is employed with both rising and falling edges to reduce the operating frequency and easing analog design and

[4]Although correct, using the term *non-return to zero* to identify this signalling method is somewhat misleading, or at least inaccurate, the reason being that in all the other more or less common signalling schemes (e.g. PAM-2^N) the transmitted symbols do not return to zero. This notwithstanding, NRZ is predominantly used in the literature to refer to PAM-2 only.

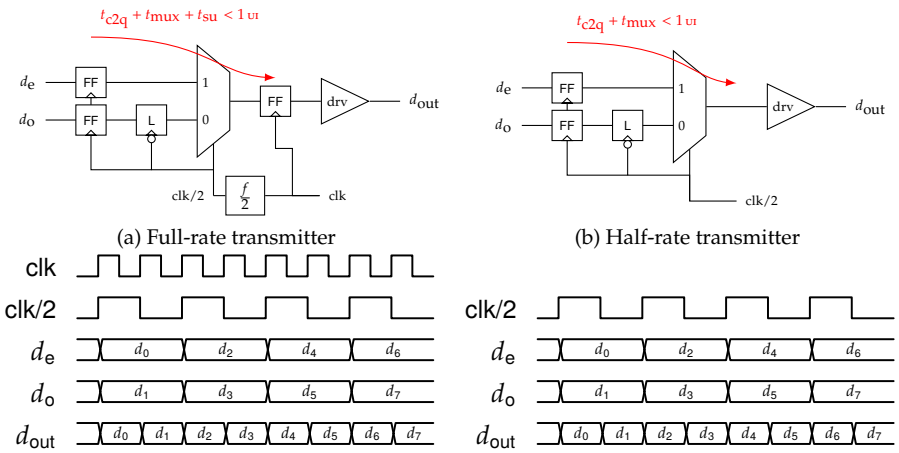


Figure 1.21: Schematic representation of the different block diagram and timing between full- and half-rate architectures (a and b, respectively), focussing on the last stage of serialization and on the timing of the critical path (shown in red).

timing constraints, as shown schematically in Figure 1.21, while resorting to more complex clock distribution architectures to guarantee a correct 50 % *duty-cycle* of the divided clock; or even *quarter rate* when a clock divided by four is exploited to drastically reduce the working frequency [25].

Baseline Wander and Encoding

As in practical applications transmitter and receiver may come from different manufactures and be fabricated in different technologies, the designer has to face the challenge of establishing a shared reference and/or common-mode voltage between them [8, 48]. In such cases, independent biasing at either end of the link is enabled by series capacitors performing AC coupling, although its combination with the termination resistors (see Section 1.3.2) forms a first-order high-pass filter which is responsible for loss in the low-frequency components of the transmitted signal, causing it to drift whenever long sequences of CIDs are sent, a phenomenon know as *baseline* or *DC wander* and depicted in Figure 1.22 [8, 48].

Intuitively, one may increase the capacitor's size in order to limit the signal's decay for CID sequences of a certain length, but for practical cases this would imply an unnecessarily large capacitor at tens of Gb/s data rates, which is rather impractical in deep sub-µm nodes [48]. Alternatively, the transmitted sequence may be digitally scrambled by performing XOR with a pseudorandom bit sequence (PRBS), thus avoiding the presence of long CIDs unless the two sequences coincide, which is the more unlikely the longer the PRBS is [8].

Often, a good compromise between the two methods above is to encode the transmitted data in order to reduce or eliminate the probability of long CID sequences. Popular solutions involve the use of NbMb *line codes* (or *transition codes*) such as 8b10b, 64b66b or similar, which deterministically map N-bit words into M-bit words preventing CIDs beyond an upper limit, at the cost of introducing an overhead onto the required data rate (e.g. 25 % in the 8b10b case, where 6.25 Gb/s

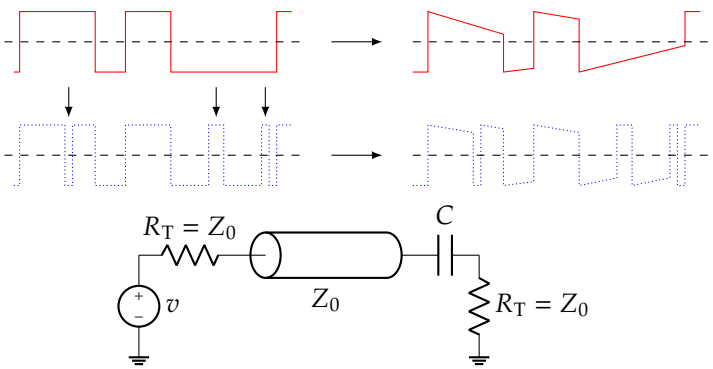


Figure 1.22: Baseline wander caused by placing a decoupling capacitor in series with the termination resistance (solid red line) and its mitigation employing a line code (dotted blue line) to increase the occurrence of transitions (shown by the arrows above the transmitted sequence).

are required to send useful data at 5 Gb/s) [8]. As a sidenote, by providing frequent toggling such transition codes aid the locking procedures in clock-recovery units (see Section 1.3.1) [13].

Serializer

Despite being conceptually easy to understand, since it can be regarded as a simple cascade of multiplexers (MUX) and registers, its design is rather critical for the transmitter since its last stages have to work at high frequency (often at the maximum data rate supported by the HSSI, if full rate) and provide retiming for the following stages through high-speed flip-flops (FFS) in order to ensure synchronization with the local clock [1, 46].

The amount of parallel data that a serializer handles determines its size and power consumption; in turn, it also determines the frequency at which the digital macro (that provides the parallel, possibly encoded, data to the transmitter) is clocked, hence its power consumption and design constraints. Therefore, the choice of serializer width is a system-level decision affecting many aspects of a transceiver's design, and has to consider the aforementioned trade-offs.

Driver

The circuits that drive the transmitted data into the channel are critical from many points of view: Their bandwidth and linearity affect signal quality and may introduce ISI even before reaching the channel; their topology determines the voltage swing across the channel, power consumption, *power supply-induced jitter (PSIJ)* sensitivity and termination impedance; their size contributes to determining their power consumption and possibly requires the implementation of pre-driving stages to guarantee signal integrity unto the driver.

From a system-level point of view, drivers are chosen depending on the voltage swing required by the target applications and standards, whereas they concur to architectural choices such as serialization ratio since they can embed the final multiplexer in order to reduce the total length traversed by the full-rate data.

1.3.5 Receiver

The receiver is responsible for the correct reception of the incoming data, a task that mainly involves the equalizers in order to cancel ISI and the clock recovery unit for synchronisation of the receiver clock to the incoming data, two blocks that will be treated in more detail in Section 4.2.2 and Appendix A, respectively. *Slicers* take decisions on the incoming signal by comparison with one or more threshold voltages (depending on the signalling scheme) to determine the received symbols; such a sequence is then deserialized and typically transferred to the digital macro for further processing, e.g. line decoding such as the aforementioned 8b10b or *forward error correction* (FEC).

Analog Front End

The received signal is immediately conditioned by the *analog front end (AFE)*, i.e. one or more cascaded stages that take care of performing amplification, noise filtering and often linear equalization through a CTLE [45]. This is an important block in the receiver, as it sets the common mode and the differential swing that will be used for further equalization and symbol detection; moreover, its noise performance and linearity are critical to determine the quality of the eye diagram and, as a consequence, the voltage margin at the slicers, hence the BER.

The summing node of the DFE can be regarded as the last analog circuit in the signal path before the slicers, so that most of the design requirements for the AFE have to be adopted for its design as well. Tap count can be a limiting factor in the receiver design due to the capacitance that its output node is required to drive, which mandates high power consumptions in order to maintain a fast settling output, although summing node architectures that relax this constraint exist [8].

Slicers

Slicers are clocked voltage comparators that are used to determine the received symbols by comparing the input signal voltage against one or more (e.g. for PAM-4 signalling) threshold voltages. Typically, their operation can be divided in phases determined by the clock: *Reset* is performed to pre-charge internal nodes at predefined voltages so to cancel or reduce memory effects that would make the current decision depend on past ones; *observation* or *amplification* marks the beginning of the comparison, when the (usually differential) input voltage difference is slowly amplified; finally, *regeneration* takes the amplified voltage difference and quickly saturates the comparator's output to the corresponding logic level through positive feedback.

Even with the brief description above, one can notice how slicers are prone to become speed bottlenecks in the receiver, since they are required to complete all their operating phases at high speed (dictated by the receiver's clock). Implementing half- or even quarter-rate receivers alleviates speed requirements for the slicers, although proportionally increasing the number of comparators that have to be instantiated. Other relevant figures of merit for slicers are related to the noise that is integrated during the observation phase and, more importantly, to the input-referred voltage offset, which has to be minimised (either by design or through calibration) in order to reduce the amount of wrong decisions that it causes.

1.4 Current Trends and Future Developments

As the rise of new and more data-demanding sectors, or simply the evolution of old applications, is driving HSSI requirements to soar ever higher (as outlined in Section 1.1), new challenges arise and new solutions are devised to tackle them. The following is a brief overview of a few trends emerged in the last decade that promise or have the potential to be game changers in the field, even though digital signal processor- or analog-to-digital converter-based (DSP- or ADC-based, respectively) HSSIs already account for a relevant share in the recent literature [7]. However, the list below is not a thorough assessment of all novelties in the field, as it is outside the scope of this thesis to provide a complete review of such trends.

Digital Signal Processor-Based HSSIs

The shift towards DSP-based wireline transceivers has already started in the literature and was mainly motivated by the need for improved equalization capable to tackle high-loss channels without the complexity of mixed-signal design. In principle, all that is required from such devices is simply to digitize the incoming signal with a fast analog-to-digital converter, transfer the received digitised voltages to the digital core and let it handle equalization, decoding, FEC, etc [4, 8].

In such a way, design complexity shifts to the digital domain, thus enabling portability across different technology nodes, power and area improvement due to scaling and easier verification, while being able to implement more complex modulation and detection schemes, and (relatively) easily deal with equalizers having large tap counts [4, 8]. In fact, DSP-based HSSIs enabled simple implementation of receive-side FFE, since noise amplification and analog delays are no longer a concern; moreover, the lack of feedback paths make parallel design of FFEs rather straightforward allowing, contrary to digital implementation of DFEs, relaxed timing margins with low power consumption, so that the trend is to implement more FFE taps and fewer DFE taps [4, 8].

The main drawbacks of such architectures arise from the overall power consumption (dominated by the DSP) and from the challenges in designing high-speed ADCs [4, 49]. The former limits such devices to applications where performance (e.g. data rate or channel loss) is the major concern, whereas typically conventional mixed-signal wireline transceivers would be too complex or just unfeasible; while the latter is an active area of research that attempts to trade off power consumption, area occupation and speed to achieve the target reception quality [4].

Specifications on the application determine the required ADC resolution, which is usually 5 to 6 bits and 7 to 8 bits for PAM-2 and PAM-4 signalling, respectively, although the effective number of bits is 1 to 2 bits lower [4]. The most widespread ADC architectures are the *flash* ADC, composed of a string of 2^N resistors and comparators capable of digitising the input voltage in a single clock cycle, although at the cost of large area consumptions; and the *successive approximation-register* (SAR) ADC which implements a binary search conversion over multiple clock cycles with reduced hardware requirements, thus trading off speed with area [4].

In order to relax timing requirements, ADCs in DSP-based HSSIs are typically time interleaved to allocate more conversion time to each individual ADC instance. This mandates that a track-and-hold circuit precedes each ADC to maintain its input voltage constant during conversion, hence adding design challenges that are commonly addressed by implementing *two-stage* architectures as that of Figure 1.23,

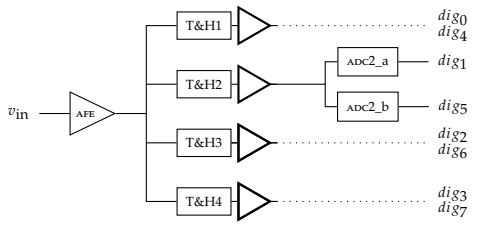


Figure 1.23: Simplified example of an 8-way time-interleaved ADC showing the working principle of two-stage architectures; T&Hs are track and hold circuits, the second T&H stage is embedded in the ADCs.

where each circuit in the first stage drives multiple second-stage track-and-holds which are activated sequentially to feed a constant voltage to the following ADC stage [4]. This greatly reduces timing constraints, but increases area consumption and complicates synthesis and distribution of the various clock phases that are required to properly drive in sequence the various blocks [4].

Other challenges peculiar of ADC-based implementations are caused by various non-ideal effects that impact the conversion system: The analog front end is split among the different parallel paths, so that gain and/or bandwidth mismatches affect conversion in different ADCs; clock skew among different track-and-holds or ADC stages and jitter degrades the eye width; comparators' offset modifies the digitised value corresponding to a certain voltage and their variation on chip implies that the same voltage will be digitised differently in different ADC stages; and, finally, the ADCs' own non-linearity also concurs in degrading conversion performance [4]. This mandates calibration techniques to compensate all the aforementioned impairments so to render all the various subchannels' transfer functions as equal as possible, as well as modelling techniques capable of efficiently taking into account each subchannel's non-ideal effects to evaluate the performance of the transceiver [4].

Another novel challenge introduced by DSP-based HSIO is related to the ADC quantization noise, which has to be accounted for, especially to consider how it is shaped by digital processing and how it consequently affects equalization and the resulting signal quality. It turns out that an equalizer's architecture with defined tap weights can be used to determine the probability density function of quantization noise at its output, thus allowing to convolve it with other noise sources when modelling statistically the performance of the receiver [4].

Then, implementing equalization faces its own challenges related to the digital implementation, mainly due to meeting timing margins and taking full advantage of digital design strategies: FFE can be implemented in a parallel and pipelined manner to relax timing constraints, which in turn allow for power scaling techniques to improve power efficiency; instead, presence of feedback paths in DFEs makes its digital implementation harder to optimise, although *look-ahead* architectures merged with conventional loop unrolling can be used to allow partial pipelining at the expense of larger area consumption [4]. As a consequence of such differences in digital equalizers implementation, the current trend is to adopt large tap-count FFEs with only a few taps handled by the DFE [8, 49].

Alternative Signalling Schemes

Communications at ever higher speeds require transceivers to operate with very high frequency clocks even when transmitting more than one bit per unit interval (e.g. a quarter-rate PAM-4 transceiver targeting 224 Gb/s standards would require a 28 GHz multi-phase clock and would be affected by channel ISI at 56 GHz Nyquist frequency), thus increasing circuit and system complexity [11]. Increasing PAM order and/or the architecture's rate alleviates clock frequency and channel loss issues, but the former brings in higher ISI sensitivity due to the reduced spacing among voltage levels, while the latter requires more complex and power-hungry clocking circuitry and equalizers [11, 49]. Therefore, in order to ease implementation of ultra high-speed wireline transceivers, various alternative signalling schemes have been proposed over the years, all targeting methods to improve spectral efficiency to fit more bits per unit of time into the same channel, two of which are briefly described below.

Multi-tone signalling over copper media mimics the ability of traditional wireless communications to transmit data over a high-frequency carrier, especially by simultaneously transmitting multiple narrowband signals all having different carrier frequencies. This has the main advantage of heavily reducing the impact of ISI, because while traditional wireline signalling excites all channel's transfer function frequencies up to the Nyquist frequency, thus experiencing different attenuations that depend on the data sequence being transmitted (i.e. ISI), single narrowband signals are simply attenuated (at first order) by the channel loss corresponding to its carrier frequency, hence greatly simplifying equalization, which is performed in baseband [49, 50].

Such a technique allows multiple sub-channels to be defined and used, possibly to avoid notches due to impedance discontinuities in multi-drop bus architectures, each supporting a specific modulation (e.g. different *quadrature-amplitude modulation* orders) so to avoid degrading the constellation when signal-to-noise ratio becomes too low at certain carrier frequencies [49, 50]. This requires a paradigm shift in designing the transceiver, as the architecture is conceptually very similar to traditional wireless circuits, e.g. by having local oscillators for de/modulation, filters to separate the various sub-channels, etc., or perform the corresponding equivalent operations when implemented as a DSP-based transceiver [49].

Chord signalling attempts at improving spectral efficiency by considering that *ISI sensitivity* (defined as the ratio between the maximum separation between voltage levels over the minimum one: It equals 1 for PAM-2 signalling, but grows to 3 for PAM-4) of PAM-N signalling of order greater than $N = 2$ increases, so that the corresponding link margins do not necessarily improve [11, 51]. From a different perspective, ISI sensitivity is larger when the distance from any voltage level to all the comparators' thresholds that are required to determine the received symbol varies;[5] therefore, ensuring that slicers always have to deal with binary signals is capable of reducing ISI sensitivity [11, 51].

This is achieved by encoding multi-bit symbols over multiple wires so that, when properly considered together, lower-rate binary signals (on each single wire) can transmit the same amount of data as faster conventional signals while having less ISI sensitivity and being subjected to less ISI, although at the cost of increased

[5] PAM-4 employs three thresholds to discriminate among four different voltage levels; as a consequence, e.g. the voltage level at $V_{DD}/3$ is $V_{DD}/3$ away from the thresholds located at $2V_{DD}/3$ and 0 V, but is V_{DD} away from the $-2V_{DD}/3$ threshold.

pin count and non-conventional transmit and receive architectures [11]. However, if *pin efficiency* is considered as a measure of the number of wires that are used to transmit a single bit, PAM-2 turns out to have an efficiency of only 0.5 (one bit over two wires in order to gain immunity to common-mode noise), whereas by sending $N-1$ bits over N wires chord signalling enhances such efficiency while maintaining immunity to common-mode noise and cross-talk typical of conventional differential signalling [11, 51].[6]

1.5 Context and Aim of the Thesis

As was mentioned in Section 1.2.1, electronics in the automotive environment is going through changes and its role is shifting towards the core of the car's functionalities and performance. As such, it is required to achieve performance (e.g. in terms of data transfer rate or processing power) that are comparable to what consumer electronics reached one or two decades ago: For the sake of consistency with the topic of this thesis, the trend of data rate in consumer and automotive electronics over the years is reported in Figure 1.24, showing such a performance lag [52].

Obviously, electronics development for automotive applications can greatly benefit from ICT and consumer-electronics standards, for the simple reason that no one is willing to spend time and money to "reinvent the wheel", but many of the existing technologies must be rethought to meet the stringent demands of the automotive industry: The requirement for performance, safety, security and usability must all be met without any trade off, and expectations are very high for proficiency and quality, hence setting the focus on features which in consumer electronics would not be considered necessary [27, 53].

The above implies that automotive electronics design at the circuit level cannot focus solely on performance, be it either data rate, latency or energy efficiency, but it has to achieve this while complying to stringent specifications regarding operating range, hence ensuring error-less operations in a wide variety of situations and especially in harsh environments. Such conditions are usually regarded as *process, voltage and temperature (PVT)* variations that the circuit has to withstand without failures: Typical automotive standards require electronics to work over all process corners[7], ±10 % voltage variations of the nominal supply voltages and temperatures ranging from −40 °C to 125 °C [53].

The above context of high-speed communications in automotive applications is the focus of this thesis, with the aim of contributing to bridge the gap between consumer and automotive electronics while complying to the requirements of the latter. Such research work was performed in collaboration with *Infineon Technologies Villach* (Austria), which has been developing such systems in the past years and provided tools, software, expertise and measurement equipments to carry out the work. The target was next-generation automotive standards such as PCIe and MIPI A-PHY, both demanding 16 Gb/s data rates over high-loss channels; considering the work that was already done at Infineon Technologies [54], three main areas of interest have been deemed as requiring further investigation to achieve this:

[6] Actually, PAM-2 signalling is the simplest case of chord signalling with $N = 2$; by increasing N, ISI sensitivity and common-mode noise immunity are maintained while pin efficiency improves [11].

[7] A well known concept in integrated-circuit electronics, process corners are characterisations of the worst-case speeds of PMOS and NMOS devices that may result from fabrication and allow simulations to take such non-ideal behaviour into account. They are usually denoted as SF (Slow-Fast), FS, SS and FF, with the first letter referring to NMOS and the second to PMOS.

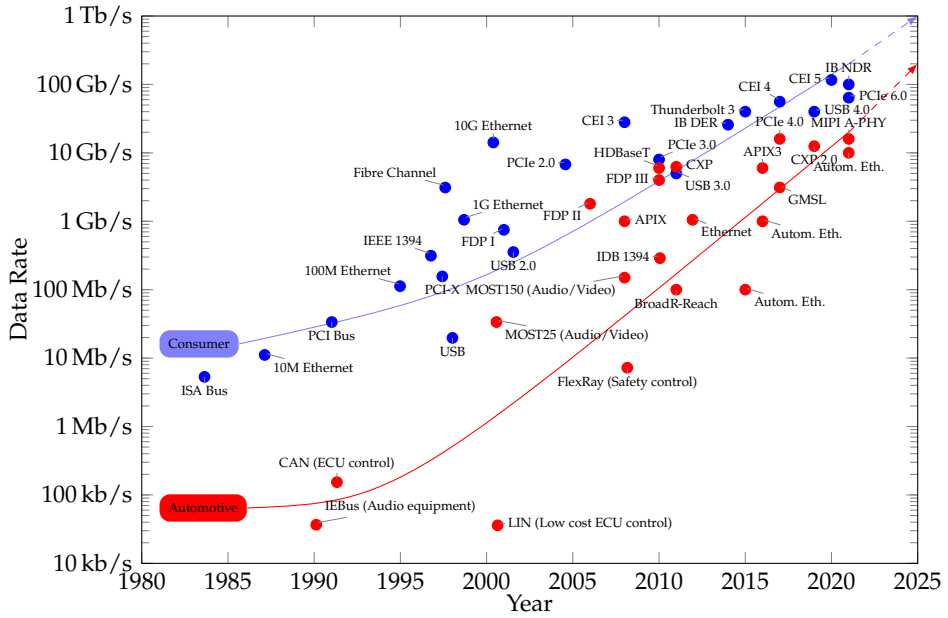


Figure 1.24: Compared trends of consumer and automotive HSSI standards. (Adapted and updated from [52].)

- Fully-adaptive equalization is widespread in consumer applications and, as data rates grow higher, its relevance is increasing also in the automotive market in order to cope with different channel losses in different applications (e.g. two instances of the same HSSI employed to communicate with a nearby processor or with distant sensors in the car), while tracking the effect of temperature changes and compensating process variations, all this without requiring any tuning by the final user;
- Clock- and data-recovery units capable of achieving good performance at high data rates are necessary to guarantee low BER in automotive applications requiring error-free operations for safety reasons;
- Moving to a half-rate transmitter (from the full-rate one previously developed in the collaboration between the University of Udine and Infineon [54, 55]) is very important when shifting to 16 Gb/s to reduce power consumption while easing clock distribution and timing margins.

These topics are investigated and developed in the following of this thesis, with the first four chapters focussing on fully-adaptive equalization: After developing the tools and theoretical framework to analyse generic HSSIs in Chapter 2 and fully-adaptive equalization in Chapter 3, the implementation of integrated fully-adaptive equalizers is presented in Chapter 4, while Chapter 5 reports the laboratory characterisation of the integrated SerDes with fully-adaptive equalization. Chapter 6 describes the development of the analog/mixed-signal 16 Gb/s half-rate transmitter and, eventually, Chapter 7 concludes this thesis by summarising the results that have been achieved and looking at possible further work. Appendix A provides some basics on jitter in HSSIs and a model to predict jitter in CDRs.

Chapter 2

Modelling ISI and Equalization in Wireline Transceivers

2.1 A Brief Overview on the Modelling of Wireline Transceivers

The increase in data rate and performance requirement that the field of wireline transceivers has witnessed in the past few decades has driven the demand for accurate modelling techniques capable of predicting HSSI behaviour. SPICE-like circuit simulators are an obvious choice for analog/mixed-signal circuits, as they allow for precise solution of complex systems taking into account advanced device models and non-linear effects, although runtime increases with circuit size and required accuracy; additionally, in order to comply with modern standards, high-speed wireline transceivers have to demonstrate operation at BER in the order of 10^{-12}, which can be achieved only by simulating a proportionate number of transmitted bits [56]; moreover, the aforementioned approach requires the transistor-level circuit (or, even better, the post-layout parameters) of the system to be simulated. This is not an optimal choice for designers aiming at defining architectures, assessing requirements due to specifications and estimating BER performance with different channels *before* starting the actual transistor-level design.

The considerations above make circuit simulations for such systems very intensive from a computational standpoint and extremely time consuming, so that many simulation frameworks have been developed over the years to improve computational performance while keeping the desired level of accuracy for HSIO modelling [57]. Many analysis techniques considering worst-case or peak distortions/noises are relatively straightforward and easy to implement, but tend to either underestimate or overestimate performance, hence motivating the development of *statistical* approaches to reduce the computational time/burden while providing accurate information on the desired figures of merit [57].

Commonly, ISI models consider the convolution of the transmitted symbols sequence and the channel's sampled pulse response (refer to the following Sections for further details) in a statistical fashion to determine the probability distribution describing the impact of ISI on the received signal [45, 57–59]. The result of such a convolution for various symbol sequences is processed to obtain a *probability density function (PDF)* of ISI at the considered sampling point, while iteration of the procedure for different points along the UI yields the statistical eye diagram [58,59]. Moreover, the same approach can be extended to include various circuits of HSSIS

(e.g. the receiver's front end) simply by considering how they modify the effective channel's pulse response [45].

The main problem in the aforementioned approaches is how to efficiently compute convolutions that have to take into account many symbol sequences to produce a useful output. In [58], all possible bit sequences spanning all relevant ISI cursors are computed and then combined with the channel's step response through a *Markov convolution* working on PDF states, thus allowing inclusion of FFE by a properly-defined mapping matrix. Another approach computes *transition PDF* (voltage distribution at the sampling point caused by pre- and post-cursors' ISI and jitter) for the various symbol transitions occurring at different times in the data stream, then properly convolves them one by one according to previous and future symbols, eventually obtaining a set of PDFS for all possible symbol sequences [45,59]. A different technique is employed in [56], where worst-case eye opening at each sampling point is found by considering one past/future symbol at the time and selecting the worst one in terms of ISI impact.

In order for single-bit response techniques to provide accurate results the system has to be linear time invariant, which generally does not hold for single-ended HSSIS that feature asymmetric rise and fall times or for relatively highly non-linear blocks [56, 59]. This can be dealt with by employing step/edge responses instead, which may be computed by considering various amounts of past bits and thus their effect on the current edge [56, 59]; moreover, this allows accounting for fully-uncorrelated jitter by altering the delay between rising and falling edges [58].

Additionally, various noise sources (crosstalk, thermal noise, etc.) can be easily included in statistical models by convolution with the resultant ISI distribution if they are mutually independent [45].

Another major topic is the inclusion of jitter in such models. One option is to separate a symbol's pulse in jittered and non-jittered components (the latter yielding information on ISI as described above), the former being modelled as pulses of different duration centred on the symbol's edges and convolved with the channel's pulse response time shifted by half T_B [57]. Other approaches simply convolve the ISI PDF in time and the jitter distribution before computing the eye diagram [58, 59]. A different technique converts jitter in voltage noise and then estimates dual-Dirac model's parameters so that so-called *interpreted TX/RX jitter* features the same statistics as the actual jitter (characterised through behavioural simulations) [45].

As the brief (and by no means thorough) review above shows, wireline transceivers modelling takes into account many aspects and blocks of HSSIS. However, this chapter proposes a method to determine only the eye diagram of jitterless wireline transceivers, once the general system architecture is known, whereas jitter can be included afterwards, provided that its variance is given (jitter and CDR modelling will be treated in Appendix A).

2.2 Proposed Model to Determine the Eye Diagram

In the following Sections a model for the system-level assessment of high-speed wireline transceivers is proposed, capable of describing PAM-2 signalling with ideal (non-jittery) clocks and absence of non-linear blocks.

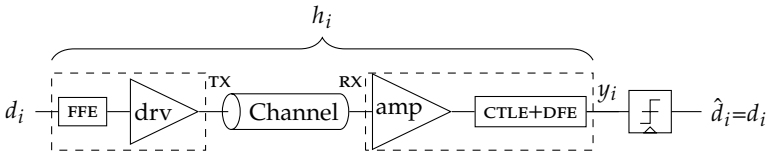


Figure 2.1: Scheme of a generic high-speed serial interface with equalization: h_i is the overall pulse response of the system composed of TX (FFE+driver), channel and RX (amplifier+CTLE+DFE).

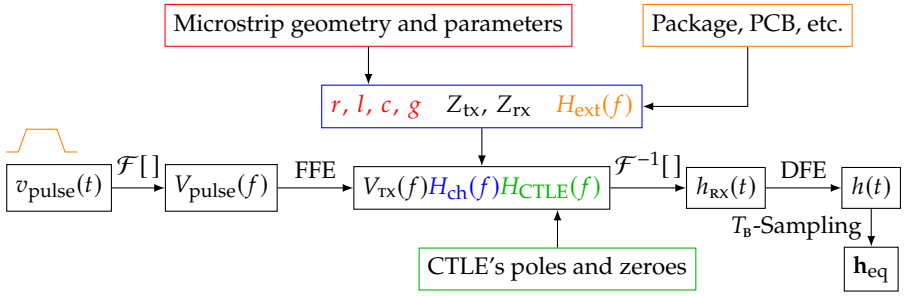


Figure 2.2: Flowchart of the procedure implemented to obtain the received pulse response. $\mathcal{F}$ and $\mathcal{F}^{-1}$ stand for the Fourier and the inverse Fourier transforms, respectively.

2.2.1 Overall System Architecture

In order to accurately predict the ISI-related system performance of a generic HSIO device, the general model depicted in Figure 2.1 and extensively described in [60, 61] is considered:

- Denoting by the subscript i the sampling instant $t_i = iT_B$ (where T_B is the bit period, i.e. 1 UI), the data sequence d_i is sent by the differential transmitter TX at bit rate $f_B = 1/T_B$ for PAM-2 signalling, optionally implementing FFE;
- the channel, whose sampled pulse response is $h_{ch,i}$, can be modelled either as two independent single-ended lines or as a coupled differential line;
- the receiver RX contains an amplifier, a CTLE and a DFE, and produces the analog voltage y_i;
- the slicer makes decisions on such a voltage ($\hat{d}_i = \text{sign}(y_i)$, i.e. $\hat{d}_i = 1$ if $y_i > 0\,\text{V}$ and -1 otherwise);
- assuming that the BER is small (either because the channel has low loss or because it is well equalized), the reconstructed data $\hat{d}_i$ is equal to the transmitted data d_i.

The numerical model implemented in Octave/MATLAB exploits a fast approach for the modelling of ISI and equalization in HSSIS, the flowchart of which is here summarised in Figure 2.2 and detailed in the following paragraphs.

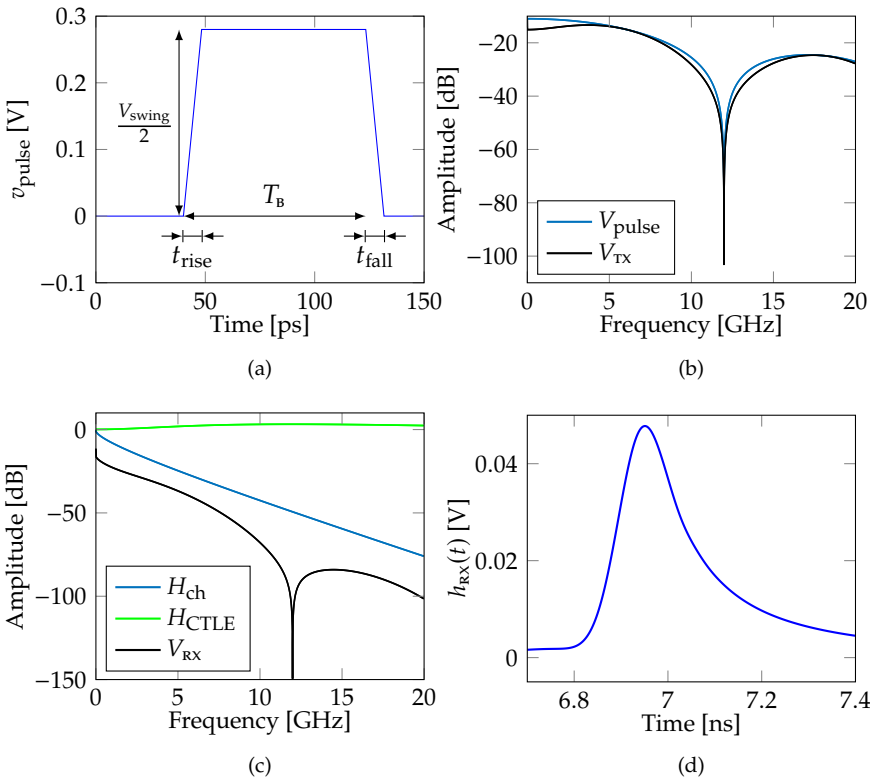


Figure 2.3: Example based on a PCIe 4.0 channel attenuating 20 dB at 6 GHz, showing the waveforms elaborated by the fast modelling tool according to the flow of Figure 2.2. (a) shows the 12 Gb/s transmitted trapezoidal pulse $v_{\text{pulse}}(t)$ with differential swing of 560 mV and 8.3 ps rise/fall times; (b) reports the corresponding Fourier transform prior to and after application of FFE ($V_{\text{pulse}}(f)$ and $V_{\text{TX}}(f)$, respectively; ≈ -4 dB on the first pre-tap, as per Equation 1.4); (c) shows $H_{\text{ch}}(f)$, $H_{\text{CTLE}}(f)$ and the result of their multiplication with $V_{\text{TX}}(f)$; (d) shows the subsequent inverse Fourier transform $h_{\text{RX}}(t)$ prior to DFE.

2.2.2 Transmitter

The idealised transmitted waveform is modelled as a trapezoidal pulse $v_{\text{pulse}}(t)$, shown in Figure 2.3a and characterised by its duration, amplitude and by the slope of its edges (rise and fall times t_{rise} and t_{fall}); such a waveform is easily Fourier-transformed assuming[1] $t_{\text{rise}} = t_{\text{fall}}$, giving

$$V_{\text{pulse}}(f) = -\frac{V_{\text{swing}}/2\text{UI}}{t_{\text{rise}}\omega^2}\left(1 - e^{-i\omega t_{\text{rise}}} - e^{-i\omega\text{UI}} + e^{-i\omega(\text{UI}+t_{\text{rise}})}\right), \tag{2.1}$$

[1]This is required to allow using the single-bit response [56, 59]: A sequence of '1' ('0') bits remains fixed at a constant high (low) level only if this condition is met.

to which FFE is applied by summing in the frequency domain weighted delayed versions of the transformed pulse itself:

$$V_{\text{TX}}(f) = V_{\text{pulse}}(f) \sum_{n=0}^{N_{\text{FFE}}-1} w_n e^{-i\omega n T_{\text{B}}}, \tag{2.2}$$

where w_n are the weights of the N_{FFE} FFE taps, subject to the amplitude constraint $\sum_{n=0}^{N_{\text{FFE}}-1} |w_n| = 1$ due to the driver architecture [45]. The effect of such an operation is shown in Figure 2.3b.

As a side note, the proposed approach considers a transmitter's impedance which is kept constant and does not change at high or low outputs, as is the case in *Input-Output Buffer Information Specification* (IBIS) models [62, 63].

2.2.3 Channel

The transmissive medium is generally modelled as a differential line, made up either of two independent microstrips (to simulate lines placed at some distance from each other as to minimise interactions) or a single coupled microstrip excited with an odd mode (to realistically reproduce differential signalling). This is then used to reproduce the salient features of any other type of transmission line, such as the target attenuation at a certain frequency or its characteristics impedance.

The microstrip features (chiefly, characteristic impedance $Z_0(f)$ and effective permittivity ε_{eff}) are computed from the line's geometry and material parameters following the approach defined in [64–70] for single-ended lossy microstrips, or extended to the coupled line case according to [71–73], and then used to extract its per-unit-length parameters $r(f)$, $l(f)$, $c(f)$, $g(f)$ considering dielectric losses and skin effect, all of which are among the main contributors to ISI [8].

Such a result can then be combined with $H_{\text{ext}}(f)$, a transfer function representing the socket or package and incorporating notch filters, which can be used to take into account discontinuities, vias, etc. in order to provide a complete description of realistic channels. $H_{\text{ext}}(f)$ can be calculated with a model of the package, e.g. in terms of parasitic resistance, inductance and capacitance, which allows a straightforward evaluation of its transfer function in terms of poles and zeroes, while the contributions due to impedance discontinuities or vias can be taken into account by fitting the features of actual measurements of the transmission line's S parameters to the transfer function of notch filters in the form

$$H_{\text{notch}}(f) = \frac{1 + 2\xi(if)/f_0 + (if)^2/f_0^2}{1 + 2(1-\xi)(if)/f_0 + (if)^2/f_0^2}, \tag{2.3}$$

where ξ is the filter's damping factor and f_0 is its notch frequency.

Using as additional parameters the driver's and the receiver's termination impedances (Z_{TX} and Z_{RX}, respectively), the transmission line transfer function is computed from the telegrapher's equations as

$$H_{\text{ch}}(f) = \frac{e^{-\gamma L}(1+\Gamma_{\text{RX}})(1-\Gamma_{\text{TX}})}{1-\Gamma_{\text{TX}}\Gamma_{\text{RX}}e^{-2\gamma L}} H_{\text{ext}}(f), \tag{2.4}$$

where $\gamma = \sqrt{(r+i\omega l)(g+i\omega c)}$ is the propagation coefficient, L is the line length and $\Gamma_{\text{TX/RX}} = \frac{Z_{\text{TX/RX}} - Z_0(f)}{Z_{\text{TX/RX}} + Z_0(f)}$ are the reflection coefficients corresponding to the transmitter and the receiver. Note that, due to the inclusion of $Z_{\text{TX/RX}}$, (2.4) takes into

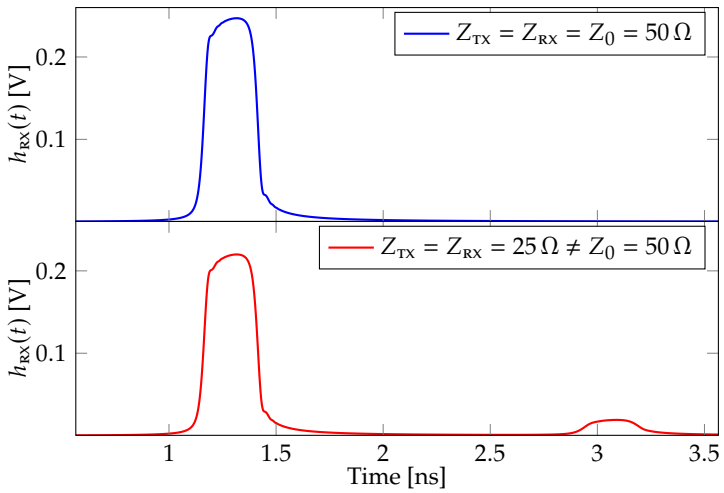


Figure 2.4: Effect of mismatch on the received pulse response: $Z_{TX} = Z_{RX} = Z_0 = 50\,\Omega$ is reported on top; below, $Z_{TX} = Z_{RX} = 25\,\Omega$, $Z_0 = 50\,\Omega$, showing reflections due to such a mismatch (also notice how the redistribution of the pulse energy into the reflection reduces its own maximum amplitude). 4 Gb/s data rate over a low-loss channel (approximately -3 dB at Nyquist frequency), with differential swing of 560 mV and 25 ps rise/fall times.

account possible non-perfect matching among driver, transmission line and receiver, which is shown as an example in Figure 2.4: The mismatch produces a reflection that contributes to ISI and reduces the main cursor amplitude.

2.2.4 Receiver

The CTLE is modelled as a rational function $H_{\text{ctle}}(f)$ characterised by the CTLE's poles and zeroes; optionally, an extraction of the CTLE's transfer function from simulations of the transistor-level HSIO can be used to reproduce more accurately a realistic implementation (and the frequency response of other analog blocks in the receiver can be similarly taken into account). The received signal associated to the transmitted trapezoidal pulse has spectrum $V_{RX}(f) \triangleq V_{TX}(f)H_{ch}(f)H_{CTLE}(f)$ (with $H_{CTLE}(f)$ including the transfer function of the amplifier/*variable-gain amplifier (VGA)* at the input of the receiver, and in general of the whole analog front end), an example of which is shown in Figure 2.3c with some of its sub-components.

The received pulse response $h(t)$ at the slicer's input is then obtained through: Inverse Fourier transform of $V_{RX}(f)$ using the procedure in [74], yielding $h_{RX}(t)$ (an example is shown in Figure 2.3d); and application of the DFE correction, performed as shown in Figure 2.5, i.e. by subtracting from $h_{RX}(t)$ rectangular pulses with amplitude equal to the tap weights a_i and centred on the sampling point, assumed to be set on the time instant corresponding to the maximum of $h_{RX}(t)$.

The procedure above implicitly assumes that the CDR has reached its steady state and is locked. Its impact on the behaviour of the HSSI is twofold: It determines the sampling point for data and edge samples (which are related to the position of the "rectangles" associated to the DFE, as mentioned above), while the jitter at its output is responsible for a reduction of BER (as will be explained in Section 2.2.6

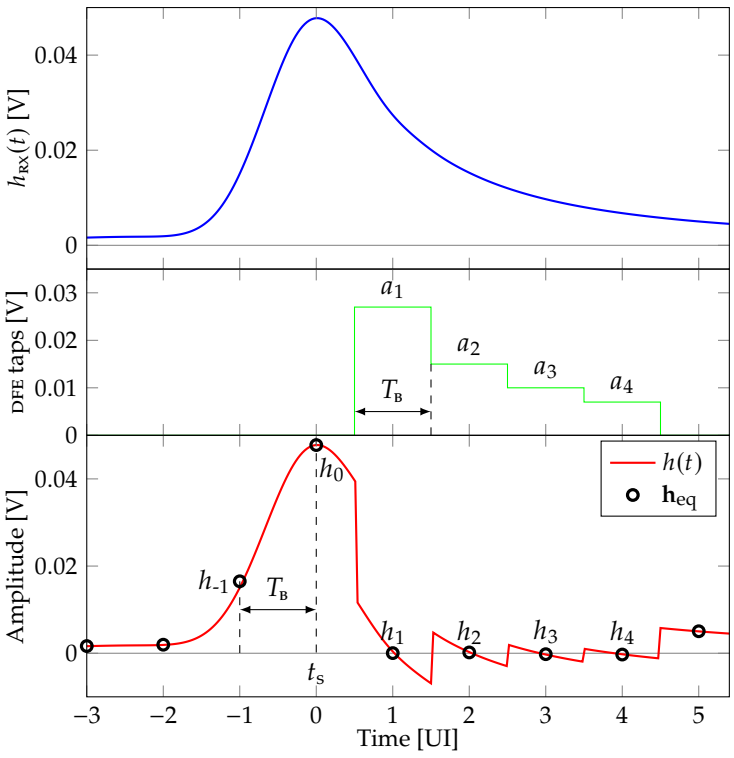


Figure 2.5: Procedure implemented in the model to apply the DFE correction to the received analog pulse response $h_{RX}(t)$ in order to obtain $h(t)$ and eventually its sampled version $\mathbf{h}_{eq}$. The DFE taps are simply modelled as rectangular pulses 1 UI wide and centred on the sampling point.

and treated extensively in Appendix A). As an alternative approach to considering the sampling point at the maximum of $h_{RX}(t)$, a CDR based on an Alexander phase detector [75] can be emulated by determining the time instants corresponding to $h_{RX,-0.5}$ and $h_{RX,0.5}$ (which are the positions of the edge samples of the CDR in a real implementation [76]) and then assume that the data sample is exactly in between. Any of the above can be selected and both of them aim at sampling as close to the centre of the eye as possible in order to reduce the probability of error.

Sampling by the bit period T_B is eventually performed in order to obtain the analog voltages (vector $\mathbf{h}_{eq} = \left[h_{-n_{pre}} \ \ldots \ h_{-1} \ h_0 \ h_1 \ \ldots \ h_{n_{post}}\right]$) that would be sensed at the input of the slicer at sampling time.

2.2.5 Computing the Eye and BER Diagrams

The performance of the transceiver is determined mainly by computing the eye diagram, constructed by folding the received signal over a time length of 1 UI, which allows to observe all the transitions that take place during operation of the serial link and their density; and by calculating the bathtub plot, which shows the cumulative distribution function of the received errors over the same time span of the eye diagram, indicating the sampling positions that result in an increased BER [8]. Both such metrics require probabilistic calculations in order to maintain

computation times low [60,77].

The proposed fast method exploits in a probabilistic fashion the well known relation

$$y_i = \sum_l h_{\mathrm{eq},l} d_{i-l}, \tag{2.5}$$

where the overall channel's pulse response and the transmitted data sequence are convolved for all relevantly non-zero values of ISI cursors to obtain the voltage level at the slicers at sampling time t_i.[2]

From the sampled pulse response $\mathbf{h}_{\mathrm{eq}}$ one can compute all possible values of the voltage y_i at the samplers, due to all the possible sequences of bits that can be sent, simply by extending Equation 2.5 into

$$\mathbf{L} = \mathbf{P}\,\mathbf{h}_{\mathrm{eq}}^T, \tag{2.6}$$

where $\mathbf{L}$ is a column vector containing such voltage levels and $\mathbf{P}$ is a *permutation matrix* which contains all the possible bit sequences of a certain length that can be transmitted. In fact, $\mathbf{P}$ is structured as a truth table: It features a number of columns equal to $n_{\mathrm{PRE}} + 1 + n_{\mathrm{POST}}$, where n_{PRE} and n_{POST} are the number of pre- and post-cursors, respectively, which can be chosen according to their relevance in the pulse response; while the number of rows is $2^{n_{\mathrm{PRE}}+1+n_{\mathrm{POST}}}$, i.e. the number of all possible sequences composed of $n_{\mathrm{PRE}} + 1 + n_{\mathrm{POST}}$ bits. In other words, $\mathbf{L}$ considers all the possible ways in which the samples of the pulse response can combine due to ISI, hence simulating observation of the received analog voltage y_i over a time span sufficiently long to gather such information. Moreover, $\mathbf{L}$ implicitly depends on the choice of the sampling instant t_i through the sampled pulse response $\mathbf{h}_{\mathrm{eq}}$.

By assuming that the eye is vertically symmetric (the behaviour of the transceiver when transmitting a '1' bit is the same as though a '0' was sent, just with a sign reversal), only the cases in which $\hat{d} = 1$ are useful for the purpose of computing the HSIO performance. By coding the '1' and '0' bits as 1 and -1 values, respectively, and keeping only the non-redundant rows, e.g. for one pre- and two post-cursors such a reduced matrix (denoted by ') reads

$$\mathbf{P}' = \begin{pmatrix} -1 & 1 & -1 & -1 \\ -1 & 1 & -1 & 1 \\ -1 & 1 & 1 & -1 \\ -1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & 1 & -1 & 1 \\ 1 & 1 & 1 & -1 \\ 1 & 1 & 1 & 1 \end{pmatrix}. \tag{2.7}$$

Equation (2.6) provides an easy way to compute the eye diagram and the bathtub plot. The eye diagram can be computed by sampling $h(t)$ at various $t = t_i \in [-T_{\mathrm{B}}/2, T_{\mathrm{B}}/2]$ and creating histograms $\mathrm{eye}_1(V, t_i)$ of the corresponding $\mathbf{L}(t_i)$; due to the fact that the eye for the '0' bit is just the mirrored version of

[2] More rigorously, Equation 2.5 should be written as $y(t_i) = \sum_l h(t_l) d_{i-l} rect(t - t_{i-l})$ to represent the continuous-time received voltage waveform at the slicers input, i.e. *before* sampling; however, using the post-sliced $h_{\mathrm{eq},l}$ instead of $h(t_l)$ is preferred in order to emphasise the fact that only the sampled values of the channel's pulse response are required for the proposed method (implicitly assuming that the slicers' aperture time is practically null).

the one for the '1' bit (as follows from the assumption of symmetry), they can be combined to obtain the overall eye diagram

$$\text{eye}(V, t_i) = \frac{\text{eye}_1(V, t_i) + \text{eye}_0(V, t_i)}{2} = \frac{\text{eye}_1(V, t_i) + \text{eye}_1(-V, t_i)}{2}. \tag{2.8}$$

The bathtub plot, i.e. the BER corresponding to a voltage threshold equal to 0 V as a function of the sampling instant t_i, is then given by the probability that a '1' bit is misinterpreted for a '0' (that is the same as the probability that a '0' bit is misinterpreted for a '1', due to the above assumption of symmetry):

$$\text{BER}(t_i) = \int_{-\infty}^{0} \text{eye}_1(V, t_i) \mathrm{d}V. \tag{2.9}$$

The overall procedure for computing the eye diagram and the bathtub plot is shown in Figure 2.6, summarising the flow described in this Section and in the following (for the inclusion of jitter).

2.2.6 Including the Effect of Jitter

A rough estimate of the effect of jitter on the receiver can be obtained by simply convolving the single $\text{eye}_{1/0}$ that form Equation 2.8 and the probability density function of sampling time t_s corresponding to the jitter component of interest $\text{PDF}_X(t_i)$ [57, 58]. As an example, considering a recovered clock at the receiver affected by random jitter characterised by the variance σ_{RJ} and having a gaussian distribution described as

$$\text{PDF}_G(t_i) = \frac{1}{\sqrt{2\pi}\sigma_{RJ}} e^{-\frac{1}{2}\left(\frac{t_i}{\sigma_{RJ}}\right)^2}, \tag{2.10}$$

the results shown in Figure 2.7 can be obtained. Figure 2.7a shows bathtub plots for different jitter values estimated by converting into random jitter the oscillator's period jitter as per Equation A.12 (refer to Appendix A for further details on jitter and CDR modelling), while Figures 2.7b and 2.7c show the effect of a sampling clock's random jitter of 6.2 ps on the eye diagram.

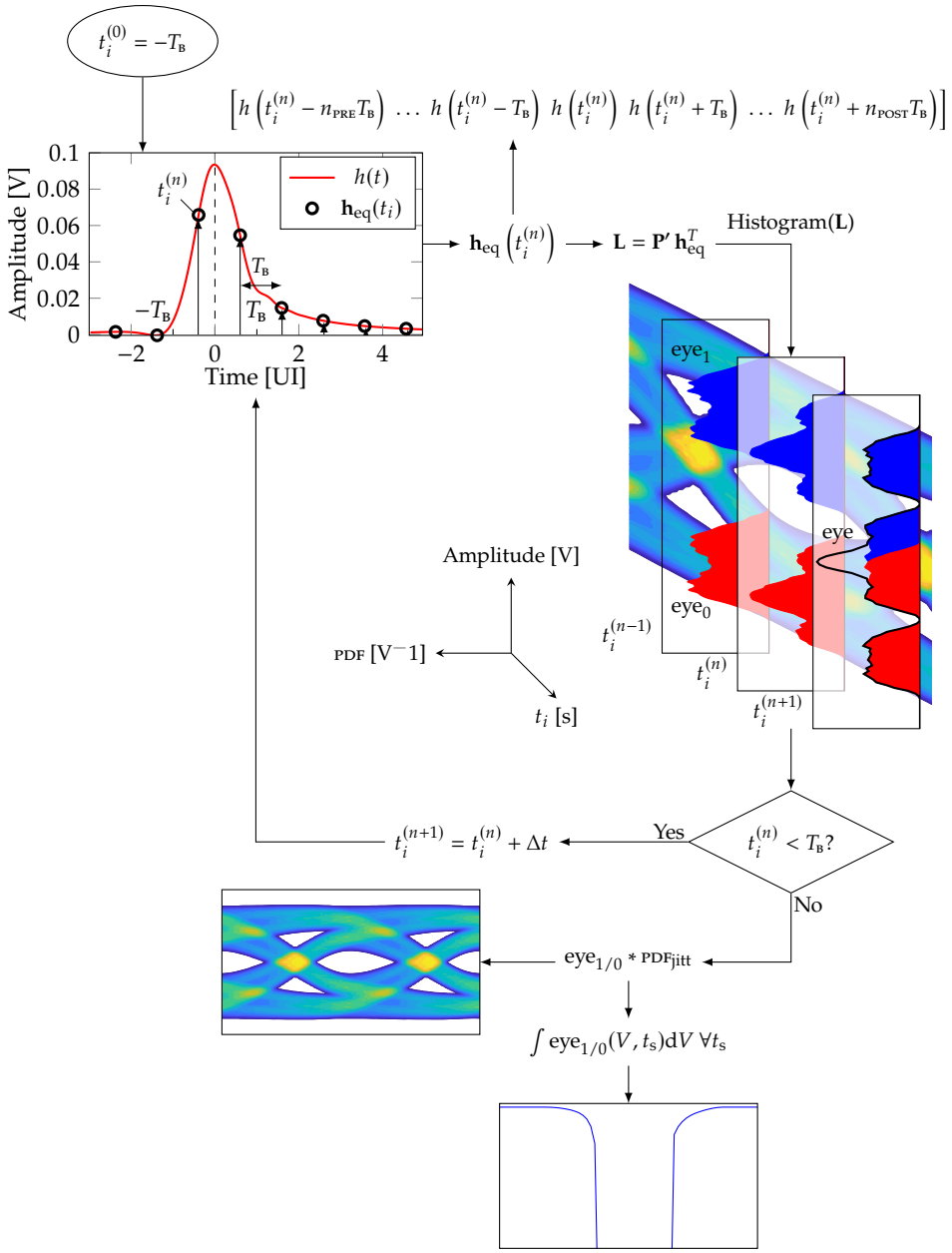


Figure 2.6: Diagram of the procedure used to evaluate the HSSI performance in terms of eye diagram and bathtub plot, as explained in Section 2.2.5. All plots and diagrams are obtained by transmitting at 12 Gb/s over a channel losing 14.3 dB at Nyquist frequency, with the FFE applying a de-emphasis of −2.5 dB; no DFE is applied for the sake of demonstrating the working principle of the proposed method. From an initial sampling point $t_i^{(0)} = -T_B$, at each iteration $n_{PRE} + n_{POST} + 1$ T_B-spaced samples of $h(t)$ are taken to form $\mathbf{h}_{eq}\left(t_i^{(n)}\right)$; at each sampling point, histograms of $\mathbf{L}$ are computed to determine the corresponding PDFs, represented in the drawing at the top right as cross sections of the eye diagram; the procedure is repeated until t_i reaches T_B and then the eye diagram and bathtub plot are computed taking into account jitter at the receiver.

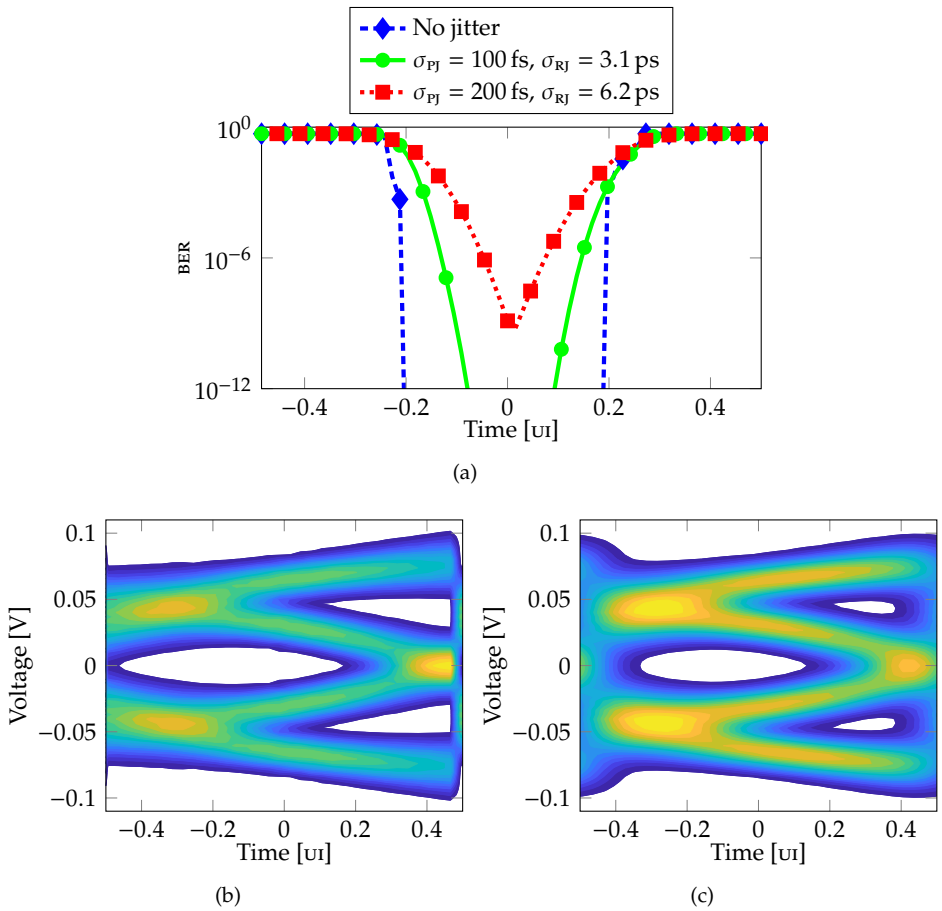


Figure 2.7: Sample results for a PCIe 4.0 channel attenuating 20 dB at 6 GHz; approximately 4 dB of FFE de-emphasis, first 4 post-cursors cancelled by the DFE. The random jitter applied at the receiver's recovered clock is calculated according to (A.12) assuming a CDR bandwidth BW_{CDR} = 1 MHz. (a) BER bathtub plots; eye diagrams with (b) no random jitter and (c) σ_{RJ} = 6.2 ps random jitter on the recovered clock.

Chapter 3

Fully-Adaptive Equalization Algorithms and Techniques

3.1 The Need for Fully-Adaptive Equalization

When dealing with equalization in the previous chapters, the frequency response from the transmitter all the way to the receiver was assumed to be known. In such a case, as was shown, optimal settings for the various equalizers can be found analytically or through analysis of the overall channel's pulse response or transfer function (see Section 1.3.3). This is not the usual situation in which most HSSIs operate, since the transmission medium is seldom known and working conditions tend to vary over time [78, 79]. Moreover, the requirement for mass-produced general-purpose products, which will be installed on different systems and affected by different operating conditions, renders a per-device tuning of the equalizers simply impossible.

As a direct consequence of the aforementioned issues, SerDes are required to automatically set their equalizers so to optimise the quality of the transmission and to adapt them to varying operating conditions, e.g. voltage fluctuations and temperature variations [78–80]; moreover, this would entail automatic compensation of effects caused by process variations.

Since the early days of telecommunications aided by digital systems, various techniques have been devised for the purpose of adaptive equalization [42, 43], chiefly *zero forcing (ZF)* [81] and *least-mean squares (LMS)* methods [82]. The latter was developed by Widrow and Hoff in 1960, and has found widespread use in HSIO applications since it was introduced by Stojanović et al. in 2005 [78]; indeed, the present chapter focusses on such a technique.

3.2 Least Mean-Squares Algorithms: Theory

3.2.1 Basic Introduction to Optimization

The equalization problem requires to determine the equalization settings which optimise some system's metric, e.g. eye height, eye width, BER, etc. Strictly speaking, this is an *optimization* problem: A solution must be found which minimises (or equivalently maximises) a well-defined *cost function* that describes the whole system or a subset of its features. Sticking to a simple analytical treatment, it resorts to finding the loci belonging to the cost function's support where its first derivative is null.

This direct approach is of course unfeasible in such cases (as that of equalization in HSSI) in which the cost function is unknown *a priori*. Therefore, an iterative procedure must be devised such that it allows the cost function to be computed as the search for the optimal solution proceeds. One of the simplest, most powerful and renowned such methods is *Newton-Raphson's* [83], which is a general procedure for solving root-finding problems and provides the basis for LMS methods: Given a function $f(\vec{x}) : \mathbb{R}^n \rightarrow \mathbb{R}$, $f(\vec{x}) \in C^2$, a quadratically-converging sequence $\{\vec{x}_k\}_{k\in\mathbb{N}}$ can be constructed by minimising at each step an approximation of $f(\vec{x})$. Once a starting point $\vec{x}_0$ is chosen such that $\vec{\nabla} f(\vec{x}_0) \neq \vec{0}$ and $\vec{x}_0 + \vec{\varepsilon}$ is sufficiently small, consider in general the 1st-order Taylor-series expansion

$$f(\vec{x}_0 + \vec{\varepsilon}) = f(\vec{x}_0) + \vec{\nabla} f(\vec{x}_0)^T \vec{\varepsilon} + \frac{1}{2} \vec{\varepsilon}^{\,T} \mathbf{H}\left[f\left(\eta(\vec{x})\right)\right] \vec{\varepsilon}; \tag{3.1}$$

if $\vec{\varepsilon}$ is sufficiently small, then the quadratic term can be neglected and, since $f(\vec{x}_0 + \vec{\varepsilon}) = 0$ by definition, Equation 3.1 can be approximated by

$$0 \approx f(\vec{x}_0) + \vec{\nabla} f(\vec{x}_0)^T \vec{\varepsilon}.$$

Solving for $\vec{\varepsilon} \triangleq \vec{x}_k - \vec{x}_{k-1}$, where k is the iteration step, such an increment with respect to the starting point $\vec{x}_0$ is found, thus allowing to write the iterative sequence

$$\vec{x}_k = \vec{x}_{k-1} - \frac{f(\vec{x}_{k-1})}{\vec{\nabla} f(\vec{x}_{k-1})}. \tag{3.2}$$

When dealing with an optimization problem, the relevant roots are those of the first derivative of the cost function, hence $f(\vec{x}) = \vec{\nabla} J(\vec{x})$, and if $\vec{\nabla} f(\vec{x})$ is positive definite then Equation 3.2 becomes

$$\vec{x}_{k+1} = \vec{x}_k - \frac{\vec{\nabla} J(\vec{x}_k)}{\mathbf{H}\left[J(\vec{x}_k)\right]}. \tag{3.3}$$

Despite its power and its fast convergence (under certain assumptions), the starting point has to be chosen accurately to ensure convergence and its main drawback lies in the computational effort that is required for calculating the second derivative of the cost function; therefore, simplification is necessary even for efficient computation on regular mainframe processors in the case of complicated functions, let alone on ICS demanding high-speed operations as in HSSIS.

The first step that can be taken in this direction yields to the *steepest descent technique*, which only converges linearly but it usually does so even for poor initial approximations. It is implemented by simply substituting in Equation 3.3 the inverse of the Hessian $\mathbf{H}[\, J(\vec{x}_k)]$ with a pre-determined, constant *step size* μ, which results in an iterative sequence which requires computing the gradient of the cost function only:

$$\vec{x}_{k+1} = \vec{x}_k - \mu \vec{\nabla} J(\vec{x}_k). \tag{3.4}$$

The effectiveness of Equation 3.4 in optimising a specific cost function can be demonstrated for the discrete-filter case (which is relevant in HSSIS), following the approach presented in [47]. Starting from the analysis developed in Section 1.3.3 with the FFE of Figure 1.17 operating on input data x (which, in this case, is

the convolution of the input data sequence and the channel's pulse response, forming the vector[1] $X_j = \left(x_j \ \dots \ x_{j-k} \ \dots \ x_{j-(N-1)}\right)^T$ at time j, with N length of the transversal filter), consider the error and the *mean square error (MSE)*[2] at the output of the FIR filter:

$$\begin{aligned} e_j &= z_{\text{des},j} - v_j = z_{\text{des},j} - X_j^T W, \\ \xi &\triangleq E\left[e_j^2\right] = E\left[z_{\text{des},j}^2\right] - 2P^T W + W^T R W. \end{aligned} \tag{3.5}$$

The MSE ξ is a bowl-shaped, quadratic function of W (the vector of the FFE taps) called *performance surface*, where $z_{\text{des},j}$ is the desired response at time j[3], $P \triangleq E\left[z_{\text{des},j} X_j\right]$ is the cross-correlation vector between input and desired response and $R \triangleq E\left[X_j X_j^T\right]$ is the (positive definite) correlation matrix of the input[4]. The so-called *Wiener-Hopf solution* can be found by zeroing $\vec{\nabla}_w \xi$, i.e. the gradient of Equation 3.5 with respect to the weight vector W, yielding

$$W^* = R^{-1} P, \tag{3.6}$$

so that, by substituting the minimum MSE $\xi_{\min} = E\left[z_{\text{des},j}^2\right] - P^T W^*$ into Equation 3.5, the MSE can be expressed as

$$\xi = \xi_{\min} + (W - W^*)^T R (W - W^*). \tag{3.7}$$

It can be demonstrated that, by rewriting Equation 3.4 into

$$W_{j+1} = W_j - \mu \vec{\nabla}_w \xi(W_j), \tag{3.8}$$

such a sequence converges and the MSE ξ approaches $\xi_{\min}$ if the largest eigenvalue of R, λ_{MAX}, is such that

$$\frac{1}{\lambda_{\text{MAX}}} > \mu > 0. \tag{3.9}$$

However, despite their apparent simplicity, both the steepest descent equations shown in Equations 3.4 and 3.8 are not suitable for implementation in HSSIS: While the former still requires knowledge of the whole cost function, the latter operates on the mean values of the input variables X_j (and $z_{\text{Nyq},j}$ in the general case), hence entailing that their complete statistics need to be known.

3.2.2 The Least Mean-Squares Algorithm

Instead of using the complete statistics of x to compute the actual gradient of the MSE,

$$\vec{\nabla}_w \xi = -2P + 2RW, \tag{3.10}$$

[1] In the following, all are column vectors unless otherwise specified.

[2] All lowercase variables are scalar quantities.

[3] The treatment presented in [47] refers to a discrete transversal filter employed to model an unknown continuous-time dynamic system, whose sampled output $z_{\text{des},j}$ has to be replicated and is characterised by a certain statistics. Instead, in the case of an FFE, $z_{\text{des},j}$ simply corresponds to the desired voltage value for a certain data bit: Assuming that signalling is differential and that all symbols are transmitted with equal probability, $E[z_{\text{des},j}^2] = 0$; this notwithstanding, since the only difference would be the expression of $\xi_{\min}$, such a mean value was kept in the equations for consistency with [47].

[4] Note that P and R, being expected values, do not depend on the time j.

an estimate based on measurements is used to compute the gradient of the square of a *single* error sample:

$$\vec{\nabla}_w e_j^2 = -2e_j X_j. \tag{3.11}$$

Therefore, the *least-mean squares (LMS)* iterative equation is

$$W_{j+1} = W_j + 2\mu e_j X_j. \tag{3.12}$$

If X_j is uncorrelated over time, then the expected value of the gradient estimate $\vec{\nabla}_w e_j^2$ equals the true gradient $\vec{\nabla}_w \xi$, and the weight-vector mean is convergent to the Wiener-Hopf solution of Equation 3.6. The sufficient and necessary condition for the convergence of the LMS algorithm is still Equation 3.9, although a sufficient condition easier to apply in practice is

$$\frac{1}{tr[R]} > 2\mu > 0, \tag{3.13}$$

where $tr[R]$ represents the total input power to the weights, which is larger than λ_{MAX} as R is positive definite.

3.2.3 The Sign-Sign LMS Algorithm

Delving deeper into the issues facing adaptive equalization in HSSIS, it is easy to see that even Equation 3.12 may be not so easy to implement in the digital hardware that is required to operate at such high speeds so to keep up with the ever-increasing bit rates nowadays achieved by SerDes, mainly because of the time needed to perform the multiplications between the real-valued e_j and X_j.

The solution to this issue was devised in the early days of adaptive equalization (1965) by Lucky [84] for ZF adaptive equalization, and can be implemented to simplify the digital hardware for the LMS algorithm as well [85], by rewriting Equation 3.12 into

$$W_{j+1} = W_j + \mu \,\text{sign}\left(e_j\right)\, \text{sign}\left(X_j\right), \tag{3.14}$$

where the factor 2 of Equation 3.12 has been embedded in μ and

$$\text{sign}(a) \triangleq \begin{cases} +1 & a > 0 \\ 0 & a = 0 \\ -1 & a < 0 \end{cases}, \tag{3.15}$$

for PAM-2 differential signal having either positive or negative polarity.

Such a further version is called *sign-sign least-mean squares (SS-LMS)* algorithm, it was introduced in the HSSI realm in 2005 by Stojanović et al. and will be the main topic of the following Sections, as it has become one of the most common solutions for adaptive equalization [78–80, 86]. Its working principle can be explained intuitively by noting that it performs adjustments on the weights based on correlations between the signs of the errors and the input data at time j, as will be further shown in the following Sections.

Another relevant feature of the SS-LMS algorithm is that, under appropriate assumptions regarding the statistical distribution of X_j, especially independence and zero mean, whether or not the algorithm converges does not depend on the magnitude of the step size; instead, under certain assumptions, μ affects only the speed and the radius of convergence [85].

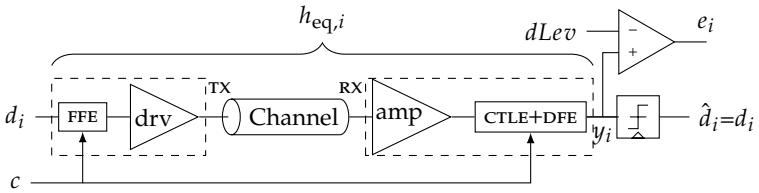


Figure 3.1: Scheme of a generic fully-adaptive high-speed serial interface: $h_{\mathrm{eq},i}$ is the overall pulse response of the system composed of TX, channel, equalizer and RX; c is a generic equalization parameter (e.g. a filter tap) on which a fully-adaptive SS-LMS loop can be applied.

3.2.4 SS-LMS Algorithms for HSSI Equalization

In order to properly understand the SS-LMS algorithm, its application to HSSI equalization is explored with the general model depicted in Figure 3.1 and partly described already in Section 2.2.1: The data sequence d_i is sent over the channel by the differential transmitter TX, which may implement FFE; the receiver RX contains a CTLE, a DFE and can adapt its sampling point. The analog voltage y_i arrives to the slicer and the bit-error ratio is assumed to be small enough that the reconstructed data $\hat{d}_i$ is equal to the transmitted data d_i. A comparator is then used to derive the error e_i by comparing y_i and the desired level $dLev$ [78], which is defined as the average value of the '1' level.

The pulse response of the system composed of TX, channel, equalizer and RX is the vector $\mathbf{h}_{\mathrm{eq}}$; therefore, the input of the slicer at time i is given by

$$y_i = \sum_l h_{\mathrm{eq},l} d_{i-l},$$

while the corresponding error is

$$e_i = y_i - dLev. \tag{3.16}$$

Full adaptation is obtained by adjusting generic equalization parameters (e.g. c in Figure 3.1 and $dLev$ itself [78]). With reference to the analysis above on optimisation algorithms, adjustments should be such as to minimize the time average $\langle e_i^2 \rangle$: Partial derivative over c yields

$$\frac{\partial \langle e_i^2 \rangle}{\partial c} = 2 \left\langle e_i \frac{\partial e_i}{\partial c} \right\rangle \tag{3.17}$$

Considering SS-LMS adaptation [42, 47, 60, 78–80, 86], the parameter c at iteration $k+1$ is adjusted as

$$c^{(k+1)} = c^{(k)} - \mu_c \left\langle \mathrm{sign}(e_i) \mathrm{sign}\left(\frac{\partial e_i}{\partial c}\right) \right\rangle = c^{(k)} - \mu_c \Delta \tag{3.18}$$

where Δ is implicitly defined and the time average is intended over the time interval between adjustments k and $k+1$.

To estimate the performance of an SS-LMS algorithm, assume that a large enough set of estimates of $\mathrm{sign}(e_i)$ is gathered in the time interval between the k-th and

$(k+1)$-th adjustments such that, in the proposed modelling approach, time averages can be replaced by ensemble averages. This is reasonable since at high data rates the data, edge and error samples are deserialized in order to be processed at a much lower speed so that large sets of samples are collected before adjusting the LMS parameters [87–90]. Using ensemble averages instead of time averages allows one to use a probabilistic approach instead of a Monte Carlo simulation of random sequences, hence increasing the simulation speed.

More specifically, exploiting the aforementioned approach, the SS-LMS update equations for the various equalizers can be easily derived by properly expressing the error e_i (Equation 3.16) [60, 78, 79] and are reported in the following.

3.2.4.1 dLev

Considering $c = dLev$, $\frac{\partial e_i}{\partial dLev} = -1$ results from Equation 3.16 and it yields [78]

$$dLev^{(k+1)} = dLev^{(k)} + \mu_{dLev}\overline{\text{sign}(e_i)}, \tag{3.19}$$

with $\Delta = -\overline{\text{sign}(e_i)}$; the error is computed only when $d_i = 1$, since by definition $dLev$ is the data level corresponding to '1'.

To gain deeper understanding of Equation 3.19, and of LMS adaptation in general, notice that $dLev$ converges when it reaches a value such that the average measured error is null; this in turn means that the received voltage y_i detected when a '1' is received is respectively larger and smaller than $dLev$ for equal amounts of time on average.

3.2.4.2 FIR filter

A transmit- or receive-side finite impulse response filter is just an FFE, whose taps are adjusted as in [78, 79]

$$w_j^{(k+1)} = w_j^{(k)} - \mu_{\text{FFE}}\overline{\text{sign}(e_i)\text{sign}(d_{i-j})} \tag{3.20}$$

where[5] $\Delta = \overline{\text{sign}(e_i)\text{sign}(d_{i-j})}$ is computed only when $d_i = 1$, since this is the condition in which the concept of $dLev$ is meaningful; this applies to all the SS-LMS equations that are described below, and will not be repeated.

A remark regarding the practical computation of the FFE taps with Equation 3.20 has to be made: As mentioned in Section 1.3.3, the maximum amplitude available at the driver's output is limited and imposes a constraint on the tap choice, i.e. $\sum |w_i| = 1$, which stems from the assumption that FFE simply redistributes power among the various taps. Therefore, when simulating FFE adaptation with the proposed model, at each iteration all taps are updated according to Equation 3.20, then normalised as $w_{j,\text{eff}} = w_j / \sum_i |w_i|$ (omitting the superscript $(k+1)$ for ease of notation) before being applied.

Equation 3.20 for the updates of FFE allows one to gain further insight into the mechanisms of SS-LMS adaptation: When the current received data bit $d_i = 1$, the

[5] The actual expression of the error in this case is $e_i = \sum_l w_l \sum_k d_k h_{i-l-k} - dLev$ due to the convolution of the transmitted data, the samples of the channel's pulse response and the FFE weights. As a consequence, the first derivative of e_i with respect to w_m yields $\sum_k d_k h_{i-m-k}$, resulting in $\Delta = \overline{\text{sign}(e_i)\text{sign}(h_0 d_{i-j})}$ under the assumption that the main cursor h_0 dominates all the others; this notwithstanding, the main cursor can be omitted in the final expression for Δ because $h_0 > 0$ always holds in PAM-2 signalling.

sign of the current error sign(e_i) is correlated to the sign of the datum sign(d_{i-j}) that was received at a time distance of j UI (earlier if $j > 0$, later if $j < 0$). If the two correlate positively (sign(e_i)sign(d_{i-j}) = 1), it means that the corresponding cursor of the channel's pulse response contributes to the error and the related FFE tap needs to be increased to counteract it; otherwise (sign(e_i)sign(d_{i-j}) = −1) the corresponding cursor does not contribute to the error or has been over-increased. The FFE taps converge when such a correlation averages to zero, which entails that the corresponding ISI contribution has been cancelled. Analogous considerations hold for all the SS-LMS equations that are considered in the following.

3.2.4.3 CTLE

Adaptation of CTLE is not straightforward to perform with SS-LMS algorithms, even though some examples can be found in the literature, despite lacking a mathematical background supporting the chosen expression for Δ in Equation 3.18 [23,79,91]; other attempts dropped adaptation based on errors at the data sampling point and focused instead on errors sampled at the signal's transitions [92,93]. In fact, with the exception of [94] performing ZF adaptation, most of the remaining approaches to CTLE adaptation rely on some sort of *heuristic* techniques: Many of them involve sweeping all CTLE coefficients to determine which one meets a predetermined target, e.g. histograms at the sampling point with largest peak values [95], equal amplitude for the signal's low- and high-frequency components [96], same signal transitions count as a reference considered with proper assumptions [97], minimum jitter corresponding to specific data transitions [98] or maximum eye opening area through an eye monitor [99]; CTLEs may be adapted by looking at rise/fall times corresponding to sequences affected differently by ISI (e.g. 0000100 vs. 0011100) and forcing the same speed for such transitions [100,101]; other methods involve estimating the energy content of low- and high-frequency components of the received signal to force a certain ratio between them by adjusting the CTLE settings [102,103]; recently, even genetic algorithms have been employed to find optimal AFE settings when the parameter space is too large for sweeping and local minima might lead to suboptimal results [104].

Commonly, CTLEs are implemented with analog filters by adding one zero with variable frequency and two poles to the channel frequency response. The poles are assumed to be fixed so that they can be embedded in the channel transfer function $H(s)$ that gives the pulse response $h(t)$. The assumption that the poles are fixed is not a limitation: These poles are part of the channel response and, if they move (due e.g. to PVT variations, corners), the adaptation loop will change the equalization parameters to compensate for that. Being the poles considered as part of the channel, CTLE is associated to a transfer function

$$H_{\text{CTLE}}(s) = c_0 + c_1 s \tag{3.21}$$

which means that a proportional plus a derivative term is applied to the received signal. While this can be easily applied to the continuous-time pulse response $h(t)$, the effect on the sampled time response is less obvious. With the help of Figure 3.2 one may notice that adding the derivative of $h(t)$ essentially shifts the maximum by approximately half a period, on channels with reasonable pre- and post-cursors. We can thus approximate the pulse response after CTLE as:

$$h'_i = c_0 \frac{h_i + h_{i-1}}{2} + c_1 \frac{h_i - h_{i-1}}{T} = h_i \left(\frac{c_0}{2} + \frac{c_1}{T} \right) + h_{i-1} \left(\frac{c_0}{2} - \frac{c_1}{T} \right), \tag{3.22}$$

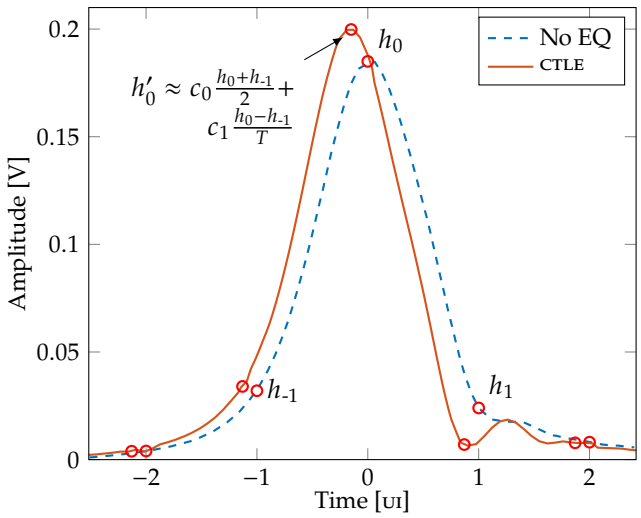


Figure 3.2: Schematic explanation of the proposed equivalence between CTLE and FIR filter using a generic channel.

which is equivalent to an FIR filter with taps $w_0 = (c_0/2 + c_1/T)$ and $w_1 = (c_0/2 - c_1/T)$. Fully-adaptive equalization can thus be performed as for FFE, eventually converting the taps w_0 and w_1 back to:

$$c_0 = w_0 + w_1 \tag{3.23a}$$

$$c_1 = (w_0 - w_1)\frac{T}{2}. \tag{3.23b}$$

Note that, depending on the sign of w_1, two cases are possible:

1. $w_1 > 0 \Rightarrow c_0 = \text{const} = 1$ due to the constraint in the FFE that $\sum |w_i| = 1$;[6]
2. $w_1 = -|w_1| < 0 \Rightarrow c_0 = w_0 - |w_1| < 1$ and $c_1 = \left(\frac{w_0}{2} - \frac{-|w_1|}{2}\right) T = \text{const} = \frac{T}{2}\ \text{s}^{-1}$ due to the same constraint (and assuming that the main cursor will not become negative).

As a consequence, when the channel is so degraded that w_1 becomes positive in order to contrast ISI, the derivative coefficient c_1 maintains a fixed value equal to half the sampling period and only the proportional coefficient varies, thus probably compromising the equalizer's capability to adapt itself to the channel and improve signal quality, since only a fixed information on the pulse's derivative will be used and only the proportional coefficient will be varied.

A 2-tap FIR filter with one pre-tap can be derived analogously by considering

$$h'_i = c_0 \frac{h_i + h_{i+1}}{2} + c_1 \frac{h_{i+1} - h_i}{T} = h_{i+1}\left(\frac{c_0}{2} + \frac{c_1}{T}\right) + h_i\left(\frac{c_0}{2} - \frac{c_1}{T}\right), \tag{3.24}$$

[6]Despite not being justified by actual hardware limitations as for the FFE case, such a constraint was maintained when adapting CTLEs with the proposed approach inherited from FFE. However, this choice can be motivated by the practical requirement of keeping the taps at "reasonable" values (i.e. so not to provide meaningless high gains), whereas actual considerations on the energy available in CTLE implementations would probably provide a constraint in the form of $\sum |w_i| = A = \text{const.}$, although this was not further investigated.

which yields slightly different CTLE weights:

$$c_0 = w_{-1} + w_0 \tag{3.25a}$$

$$c_1 = (w_{-1} - w_0)\frac{T}{2}. \tag{3.25b}$$

In this case, the consequence of the sign of w_{-1} is expressed as follows:

1. $w_{-1} > 0 \Rightarrow c_0 = \text{const} = 1$;
2. $w_{-1} = -|w_{-1}| < 0 \Rightarrow c_0 = w_0 - |w_{-1}| < 1, c_1 = \left(\frac{-|w_{-1}|}{2} - \frac{w_0}{2}\right) T = \text{const} = -\frac{T}{2}\,\text{s}^{-1}$.

Transversal filters or pre-filters can be used to add a second zero by simply considering two cascaded first-order CTLEs [94, 105, 106]:

$$H_{\text{CTLE}}(s) = c_0 + c_1 s + c_2 s^2 = (b_0 + b_1 s)(q_0 + q_1 s). \tag{3.26}$$

While the circuit corresponding to Equation 3.21 adds a zero with negative frequency, the two zeros from Equation 3.26 can have arbitrary signs. As in SS-LMS adaptation of FIR filters, each coefficient is adjusted by evaluating the correlation between the current error e_0 and a certain cursor of the received signal d_k. Therefore, assuming that the first CTLE block acts on the first pre-cursor and the second one on the first post-cursor (there would be no point in having both first-order CTLEs to work on the same cursor when equalization of the pulse response is considered), with the aid of Equations 3.20, 3.23 and 3.25 it follows that b_1 (pre-cursor CTLE) is updated depending on the correlation $\overline{d_1 e_0}$ while q_1 (post-cursor CTLE) depends on $\overline{d_{-1} e_0}$.

3.2.4.4 DFE

The DFE taps a_j are subtracted from the received analog signal (see Section 2.2.4). The derivation of the SS-LMS update equation is rather straightforward and starts by rewriting Equation 3.16 as

$$e_i = \sum_l h_{\text{eq},l} d_{i-l} - \sum_{l=1}^{N_{\text{taps}}} a_l d_{i-l} - dLev, \tag{3.27}$$

from which $\frac{\partial e_i}{\partial a_j} = -d_{i-j}$ follows. Then, DFE taps are adjusted by the adaptive SS-LMS loop as in [79, 86]:

$$a_j^{(k+1)} = a_j^{(k)} + \mu_{\text{DFE}} \overline{\text{sign}(e_i)\text{sign}(d_{i-j})}, \tag{3.28}$$

where $\Delta = \overline{\text{sign}(e_i)\text{sign}(d_{i-j})}$.

3.2.4.5 Optimal Sampling Phase

The receiver's adaptability to technology corners, voltage and temperature variations can be further enhanced by adding the capability to find the optimal sampling point at run time, which results in another degree of freedom for the whole adaptation procedure. This is done simultaneously to the clock- and data-recovery process, and its update equation is [79]

$$\varphi_{\text{shift}}^{k+1} = \varphi_{\text{shift}}^{k} - \mu_{\varphi} \overline{\text{sign}(e_i)\text{sign}(d_{i-1})}, \tag{3.29}$$

where $\Delta = \overline{\text{sign}(e_i)\text{sign}(d_{i-1})}$; φ_{shift} is considered as a phase shift (or, equivalently, as a time shift) with respect to the sampling phase computed by the CDR.

Such a technique can improve the SNR of the received signal by shifting the receiver's sampling position so that, although the main cursor is dragged away from the peak of the pulse response, one of its neighbouring cursors (e.g. the first pre-cursor) is driven close to zero at a faster rate. Additionally, this inherently avoids issues caused by a possible residual first pre-cursor that would hamper adaptation of the other equalizers, and that would otherwise require to include an estimate of h_{-1} in other SS-LMS update equations (e.g. for the DFE) [80].

3.3 Implementation of the SS-LMS Algorithm

The development of the SS-LMS algorithm in HSSIS with the target of fabricating a test-chip employing fully-adaptive equalization went through a certain number of steps: At first, a purely numerical version based on the modelling approach of Chapter 2 was implemented in Octave/MATLAB in order to consider an idealised, yet accurately-modelled system through which it was possible to gain important insight of the working principles supporting SS-LMS equalization [60]; then, starting from the aforementioned knowledge of the algorithm, a behavioural version was coded in Verilog-A in order to test the fully-adaptive machinery with post-layout transistor-level implementations of Infineon's HSSI [76]; eventually, yet another version of the fully-adaptive algorithm was implemented as a *register transfer level* (RTL) description in VHDL in order to allow digital synthesis and subsequent fabrication (described in Chapter 4).

3.3.1 Algorithm Implementation in Octave/MATLAB

The numerical model implemented in Octave/MATLAB exploits the fast tool for the modelling of ISI and equalization in HSSIS which was thoroughly described in Chapter 2 as well as in [60, 61, 107]. With reference to the analysis of Section 2.2.5, computation of the various forms of Δ in Equation 3.18 translates into

$$\overline{\text{sign}(e_i)} = \frac{\sum \text{sign}\left(\mathbf{P}'\,\mathbf{h}_{eq}^T - dLev\right)}{2^{n_{PRE}+n_{POST}}}, \tag{3.30}$$

$$\overline{\text{sign}(e_i)\text{sign}(d_{i-j})} = \frac{\sum \text{sign}\left(\mathbf{P}'\,\mathbf{h}_{eq}^T - dLev\right)\cdot \mathbf{P}'_j}{2^{n_{PRE}+n_{POST}}}, \tag{3.31}$$

where $\cdot$ is the dot product and $\mathbf{P}'_j$ is the j-th column of $\mathbf{P}'$. Given that $\mathbf{P}'\,\mathbf{h}_{eq}^T$ provides the voltage levels of all possible bit sequences of the considered length (see Equation 2.6), subtracting $dLev$ results in an error voltage for all such sequences; then, multiplying the sign of such error with $\mathbf{P}'_j$ (which contains only ± 1, see Equation 2.7) is effectively equivalent to the product with sign $\left(d_{i-j}\right)$; finally, division by the number of considered sequences $2^{n_{PRE}+n_{POST}}$ performs the averaging.

Notice the dependence of Equations 3.30 and 3.31 on $\mathbf{h}_{eq}$: Since the pulse response is modified by the equalizers, when computing the SS-LMS adjustments it has to be recomputed after each iteration of the adaptive algorithm in order to determine the new values of the ensemble averages above. Equations 2.8 and 2.9

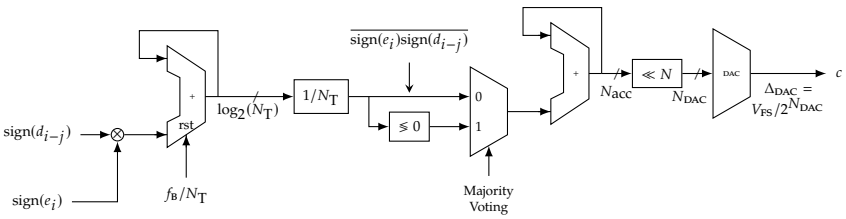


Figure 3.3: Simplified schematic of a hardware implementation of the SS-LMS loop to take into account digital precision. The first accumulator is reset every N_T clock cycles (frequency f_B), majority voting is performed by the $\lessgtr 0$ block, $\ll N$ is a shifter (divider by 2^N) and V_{FS} is the full-scale voltage of the N_{DAC}-bit DAC; the registers after the accumulators are not drawn for simplicity.

are then regularly used to evaluate the eye diagram and the bathtub plot when the SS-LMS algorithm reaches convergence.

Using Equations 3.30 and 3.31, the evolution during time of $dLev$ and of the equalizer adaptation can be estimated. Actually, what the numerical version of the algorithm provides is the evolution of the adaptation over the number of algorithm iterations only, as the simulation procedure is statistical, therefore time-unaware (or time-agnostic). In order to estimate the actual time lapse in which adaptation occurs, the bit period has to be taken into account and assumptions on the number of bits to process at each iteration and on possible averaging need to be made. This, however, would only be an approximation since the probabilistic approach assumes an iteration time so long that all the possible sequences (of a certain length, equal to $n_{PRE}+1+n_{POST}$) of transmitted bits are detected and processed, while this is unlikely to occur in real applications; moreover, such sequences might actually be reduced in number by encodings such as 8b10b or 64b66b, which eliminate some sequences that would result in severe ISI (chiefly, those containing long CIDS, as explained in Section 1.3.4).

3.3.1.1 Considerations on Digital Precision

Before presenting some simulation results, some important features of Equations 3.30 and 3.31 which affect a realistic application of the SS-LMS algorithm should be mentioned: The terms resulting from Equations 3.30 and 3.31 are real numbers in the range between -1 and 1, due to the averaging. The effect of the finite number of bits of the *digital-to-analog converters (DACS)*, which are used to adjust the equalization parameters in any hardware implementation, has to be applied in the modelling procedure *after* the evaluation of such equations and the step size μ_c of Equation 3.18 can be exploited for this purpose. The schematic of Figure 3.3 represents a simplified hardware implementation of an SS-LMS loop and is taken as a reference for the following discussion.

Direct use of the results of Equations 3.30 and 3.31 corresponds to employing DACS with an *almost infinite*, arbitrary number of levels $2^{N_{DAC}}$: In the terminology of

Equation 3.31, the update equation for a generic parameter c can be written as

$$\begin{aligned}\mu_c\overline{\text{sign}(e_i)\text{sign}(d_{i-j})} &\approx \frac{\mu_c}{2^{n_{PRE}+n_{POST}}}\sum \text{sign}\left(\mathbf{P}'\,\mathbf{h}_{eq}^T - dLev\right)\cdot\mathbf{P}'_j \\ &\approx \frac{\Delta_{\text{DAC,mat}}}{2}\sum \text{sign}\left(\mathbf{P}'\,\mathbf{h}_{eq}^T - dLev\right)\cdot\mathbf{P}'_j, \end{aligned} \tag{3.32}$$

where $2^{n_{PRE}+n_{POST}}$ determines the number of levels for a single update of parameter c, whereas μ_c defines their amplitude; since the result of the summation takes values between $-2^{n_{PRE}+n_{POST}}$ and $2^{n_{PRE}+n_{POST}}$, the update of c is in the range $[-\mu_c\ \mu_c]$ and is performed in steps of $2\mu_c/(2^{n_{PRE}+n_{POST}}-1)$, so that the equivalent DAC step size in the probabilistic model is $\Delta_{\text{DAC,mat}} \approx \mu_c/2^{n_{PRE}+n_{POST}-1}$ for a sufficient number of cursors. Referring to Figure 3.3, the number of observed bit sequences before updating c is N_T and it determines the resolution used to represent the average $\langle \text{sign}(e_i)\,\text{sign}(d_{i-j})\rangle$: The output of the first adder at the time of reset is in the range $[-N_\text{T}\ N_\text{T}]$ with granularity[7] of 2, so that the minimum step that can be made in the last accumulator is $\Delta_\text{acc} = 2/(N_\text{T}-1)$. The N_DAC-bit DAC has, by definition, a granularity of $\Delta_\text{DAC} = V_\text{FS}/2^{N_\text{DAC}}$, where V_FS is its full-scale output voltage; imposing a 1:1 match for the levels of the DAC to those of the digital circuitry requires that the input of the DAC (i.e. the output of the accumulator after division by 2^N) have granularity equal to 1 (e.g. ±1 maps to $\pm\Delta_\text{DAC}$), which means that $\Delta_\text{acc}2^{-N} = 2^{-N+1}/(N_\text{T}-1) = 1$ has to hold, so that

$$N = 1 - \log_2(N_\text{T}-1) \approx 1 - \log_2(N_\text{T}), \tag{3.33}$$

for long enough observation times.[8] Now, the DAC's input ranges between -2^{-N} and 2^{-N} (still in $N_\text{T}-1$ steps) and maps to voltages between $-2^{-N}\Delta_\text{DAC}$ and $2^{-N}\Delta_\text{DAC}$ with steps Δ_DAC wide. When emulating such a situation with the proposed probabilistic model, consider that at each iteration the output varies in the range $[-\mu_c\ \mu_c]$ in $2^{n_{PRE}+n_{POST}-1}$ steps, so that $\mu_c = 2^{-N}\Delta_\text{DAC}$ holds, i.e.

$$\mu_c = 2^{-N}\frac{V_\text{FS}}{2^{N_\text{DAC}}} \tag{3.34}$$

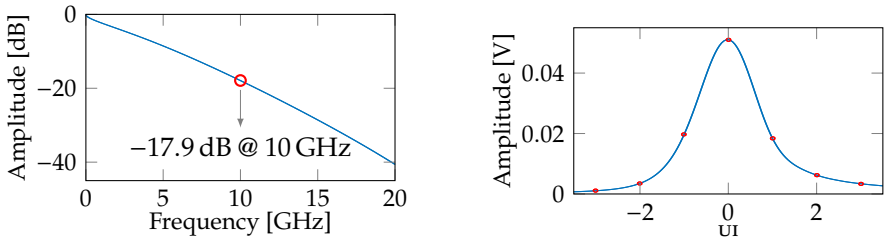
links the variation range for each iteration of the probabilistic model with that of the hardware implementation, despite with different granularities that depend on $2^{n_{PRE}+n_{POST}-1}$ in the former case and on N_T in the latter.

One can also consider hardware implementations where *majority voting* is performed on the samples collected between two adjustments, i.e. the sign function is applied to the results of Equations 3.30 and 3.31, so that the granularity of the accumulator is simply equal to $\Delta_\text{acc} = 1$: At each SS-LMS iteration the word at the input of the DAC has to be either increased or decreased by 1, regardless of the number of accumulated samples, N_T. In Figure 3.3, it means that $N = 0$ and majority voting is set to 1, whereas when considering the numerical model $\mu_c = \Delta_\text{DAC}$ has to be set and only the sign of Equations 3.30 and 3.31 taken into account.

Instead, if the DACs have a generic granularity Δ_DAC but the precision of the digital hardware implementing the SS-LMS algorithm (in particular the accumulator implied by Equation 3.18) is finer, one can simply apply quantization to the

[7] The result of the summation is 0 if half the sequences yield 1 and the other half -1; the minimum allowed unbalance occurs when $N_\text{T}/2+1$ sequences result in, e.g. 1 and the remaining $N_\text{T}/2-1$ have opposite polarity: The result of the summation is 2.

[8] Despite Equation 3.33 could easily be expressed as a multiplication, in one of the following paragraphs the division will prove to be more straightforward.



(a) (b)

Figure 3.4: Simulated (a) frequency response and (b) corresponding pulse response to a trapezoidal pulse for the channel considered in [60]. The channel's characteristics were extracted from [108] with the addition of a π-network to describe a typical package. The trapezoidal pulse was sent at 20 Gb/s bit rate, with 20 % rise time and ±0.25 V swing differential voltage.

Table 3.1: Cursors (in [V]) of the channel in Figure 3.4 after sampling.

$h_{\text{eq,-2}}$	$h_{\text{eq,-1}}$	$h_{\text{eq,0}}$	$h_{\text{eq,1}}$	$h_{\text{eq,2}}$	$h_{\text{eq,3}}$
0.0035	0.0197	0.0511	0.0184	0.0062	0.0034

$c^{(k+1)}$ in Equation 3.18 *before* using that parameter to compute the equalized pulse response, thus mimicking the effect of driving a DAC having a lower resolution than the available control word. Referring to Figure 3.3 for the consequences on the hardware implementation, this translates to $N_{\text{DAC}} < N_{\text{acc}}$: The final accumulator's output has to be shifted to the right by $N = \log_2(N_{\text{acc}}/N_{\text{DAC}})$ bits (i.e. the represented numerical value is divided by 2^N), so that the least significant bits are dropped to fit the lower resolution of the DAC.

Therefore, with proper choice of μ_c or quantization of either Δ or $c^{(k+1)}$ of Equation 3.18, one can model the finite number of bits in the DACs and in the digital hardware, hence being able to model a wide number of real situations with few, simple parameters.

The examples below showing the behaviour of the various SS-LMS loops are obtained considering a realistic channel as the one reported in [108] (*Channel 3* in that paper) connected to a package described by an LC π-network having $L = 2$ nH and $C = 100$ fF; the overall transfer function is reported in Figure 3.4a. The pulse response without CTLE is reported in Figure 3.4b, assuming a trapezoidal pulse with 20 Gb/s bit rate, 20 % rise/fall time and ±0.25 V differential voltage swing; the sampled pulse response is reported in Table 3.1 for the most relevant cursors.

3.3.1.2 dLev

As an example of convergence of the algorithm, consider the channel in Figure 3.4 without any equalization and only the loop for $dLev$ running. Results are shown in Figure 3.5 and show that $dLev$, as expected, converges close to $h_{\text{eq,0}} = 51.1$ mV (see Table 3.1) and that convergence is faster as μ_{dLev} increases. The eye diagram for the unequalized channel is reported in Figure 3.5c and shows a closed eye featuring a BER of 2 % at its centre; this notwithstanding, the functioning of the SS-LMS algorithm for $dLev$ is not compromised due to the working principle of

the probabilistic simulation approach, whereas in real applications, as defined by many standards, the serial link undergoes transmit-side calibration at start-up which helps reducing the BER at the receiver (as can be seen in the sections below). As mentioned above, Figure 3.5c also shows that $dLev$ is the average value of the '1' level at the center of the eye, as expected [78].

The results in Figure 3.5a refer to the case with an almost infinite number of bits in the DAC producing the $dLev$ signal sent to the error comparator (see Figure 3.1), and in the accumulator driving that DAC (refer to Figure 3.3). In Figure 3.5b the case where the granularity of the DAC is equal to μ_{dLev} is considered, so that only the sign of Equation 3.30 is employed (in other words the FSM, at each iteration, decides either to increase or decrease by 1 the control word at the input of the DAC, following a majority voting): Now $dLev$ oscillates between two values spaced by μ_{dLev} instead of converging to a single voltage.

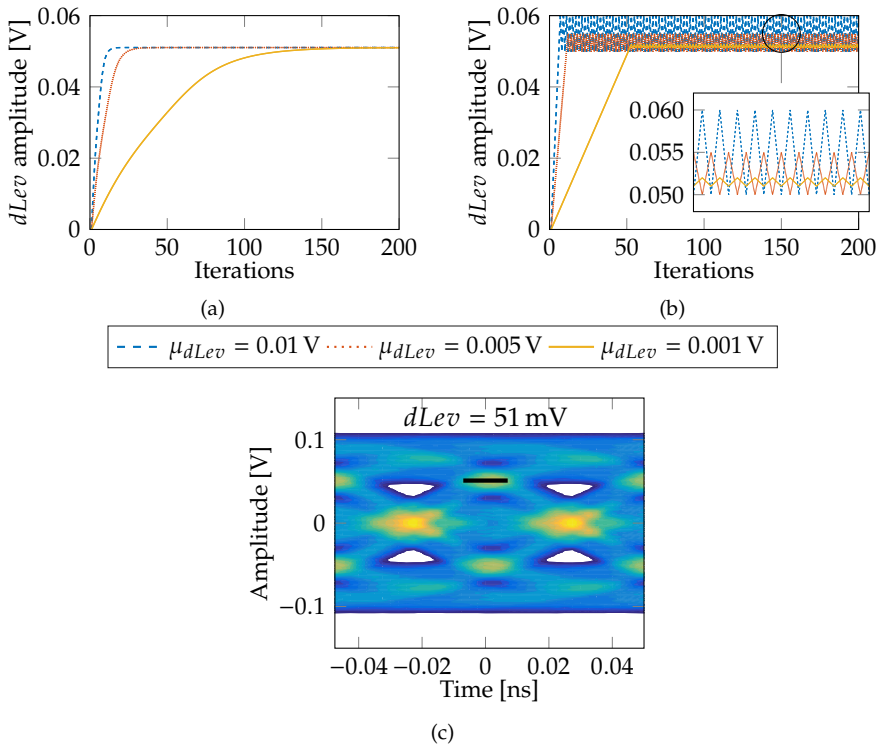


Figure 3.5: SS-LMS loop for $dLev$ for the channel in Fig. 3.4 without equalization. Convergence of $dLev$ as a function of the number of iterations in the simulation routine computed (a) assuming an infinite number of bits in the accumulator and DAC, or (b) granularity of DAC equal to μ_{dLev} and majority voting (the inset displays a magnified version of the upper portion to show the behaviour at convergence due to the different values of μ_{dLev}); and (c) $dLev$ from plot (a) shown on the (closed) eye diagram without equalization, which is characterised by a BER of 2 % at eye centre.

3.3.1.3 DFE

Sample results for a three-tap DFE are shown in Figure 3.6 for the channel of Figure 3.4. In plot (a) we see that the taps converge to the corresponding values of the pulse response, as expected from the considerations in Section 1.3.3. The equalized eye diagram shown in Figure 3.6b features a significantly-improved signal-to-noise ratio (SNR) with respect to the unequalized eye in Figure 3.5c. The SNR is computed exploiting the proposed statistical method to account for all the possible transmitted bit sequences as

$$\text{SNR} = \frac{\overline{\mathbf{L}}}{\sqrt{\overline{\left(\mathbf{L} - \overline{\mathbf{L}}\right)^2}}} \tag{3.35}$$

where $\mathbf{L}$ contains all the possible levels at the receiver, as defined in Equation 2.6, and the overline indicates averaging over the vector components.

One may wonder why $dLev$ in Figure 3.6b does not converge to h_0 as in Figure 3.5a. The explanation is that the sign-sign LMS algorithm converges as soon as $dLev$ is such that there is an equal number of y_i levels higher than $dLev$ and lower than $dLev$. In the unequalized channel, the sampled pulse response features a lot of cursors so that the levels $h_0 + \sum_{i\neq 0} \pm h_i$ form a set resembling a continuum with elements approaching very closely h_0. When we apply DFE, the first three post-cursors are almost eliminated, so that the levels become more *interspersed*: When

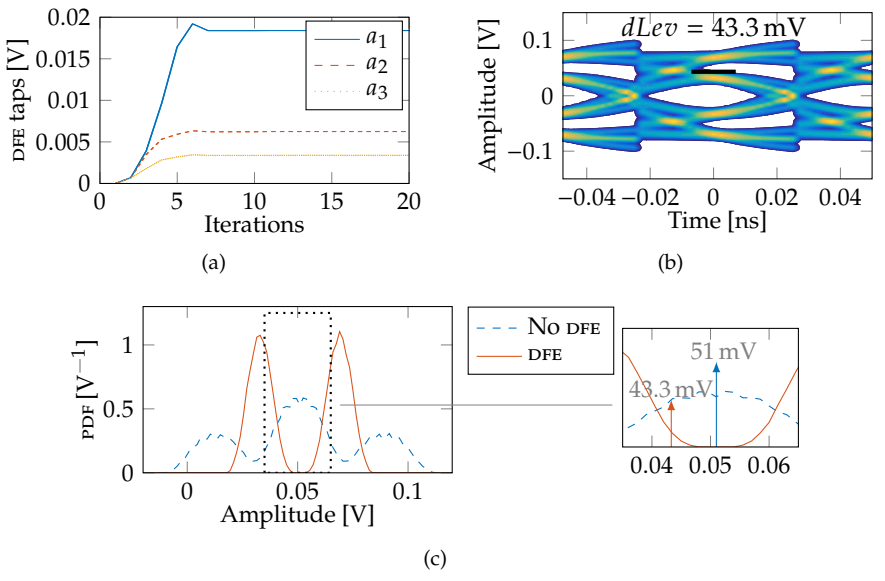


Figure 3.6: Adaptive three-tap DFE on the channel of Figure 3.4: (a) Convergence of the DFE taps a_i to 18.4 mV, 6.2 mV and 3.4 mV (with $\mu_{\text{DFE}} = 0.01$ V) as a function of the number of iterations in the simulation routine and (b) equalized eye diagram with $dLev = 43.3$ mV (with $\mu_{dLev} = 0.01$ V). Plot (c) reports the histograms of the voltage distribution at the sampling position when detecting a '1' bit without and with DFE, showing the two $dLev$s and why they differ in the two cases.

$dLev$ moves up from 0 V, it reaches a value lower than h_0 where the number of levels lower than $dLev$ equals the number of levels above it. As shown in Figure 3.6c, adaptation of the DFE taps eliminates the first three, most relevant post-cursors so that, referring to the pulse response of Figure 3.4b, the first pre-cursor becomes the dominant contributor of ISI; the voltage distribution at the sampling point effectively becomes bimodal with centres at $h_0 \pm h_{\text{-}1}$ and the two distributions are so relatively narrow that no voltage contribution appears in the neighbourhood of h_0: As soon as $dLev$ passes through the leftmost distribution, its adaptation loop sees an equal number of ISI levels on either side, $\overline{e_i} = 0$ and $dLev$ stops updating; a similar phenomenon driven by the influence of a relevant first pre-cursor was reported in [80] regarding the convergence values of DFE taps.

3.3.1.4 FIR filter

Sample results for a four-tap transmit-side FFE are reported in Figure 3.7: Plot (a) shows the time evolution of the taps, while plot (b) shows the equalized eye diagram. FFE cancels the first pre-cursor, resulting in a better SNR compared to DFE (19.1 dB vs. 8 dB at the sampling point); however, $dLev$ is reduced since the main tap is smaller than 1 V (i.e. FFE redistributes the transmitter's power to the various enabled taps). Solution to a least-squares minimization problem (refer to Section 1.3.3 and [42, 47]) involving the pulse response of the channel yields the *zero forcing (ZF)* formula for the FFE weights

$$w_{ZF} = (\mathbf{H}_{\text{ch}}^T \mathbf{H}_{\text{ch}})^{-1} \mathbf{H}_{\text{ch}}^T z_{\text{Nyq}} \tag{3.36}$$

$$= (-0.2021, 0.6017, -0.1924, 0.0037)\,\text{V} \tag{3.37}$$

(where $\mathbf{H}_{\text{ch}}$ is the matrix which rearranges the samples of the pulse response of the channel in order to describe convolution with $h_{\text{eq}}(t)$, and z_{Nyq} is the desired, ideal channel pulse response) that are very close to the steady-state values of the taps in Figure 3.7a.

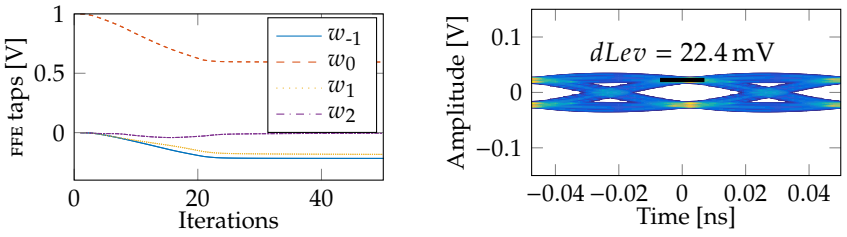


Figure 3.7: Adaptive four-tap FFE applied to the channel of Figure 3.4: (a) Convergence of the FFE taps w_i to −0.2155 V, 0.5886 V, −0.1887 V and 0.0072 V (with $\mu_{\text{FFE}} = 0.01$ V) as a function of the number of iterations in the simulation routine and (b) equalized eye diagram with $dLev$ = 22.4 mV (with $\mu_{dLev} = 0.01$ V).

3.3.1.5 CTLE

Sample results for a first-order CTLE are shown in Figure 3.8: c_0 remains close to 1 while c_1/T increases and becomes larger than the initial value 0.5 (Figure 3.8a).

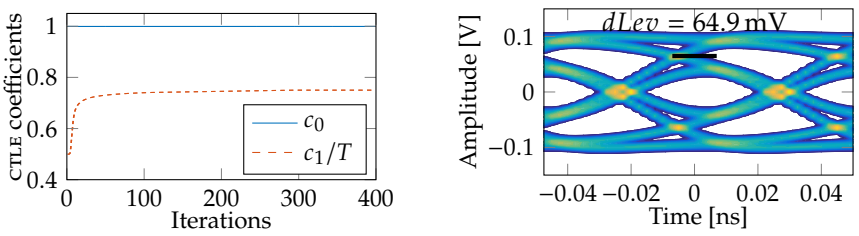


Figure 3.8: Adaptive first-order CTLE on the channel of Figure 3.4: (a) Convergence of the CTLE taps to $c_0 = 1$ and $c_1/T = 0.7506$ (with $\mu_{\text{CTLE}} = 0.05$) as a function of the number of iterations in the simulation routine and (b) equalized eye diagram with $dLev = 64.9\,\text{mV}$ (with $\mu_{dLev} = 0.01\,\text{V}$). The poles associated to the CTLE were assumed to be $w_{\text{p1}} = 20\,\text{GHz}$ and $w_{\text{p2}} = 30\,\text{GHz}$ and have been added to the transfer function of the channel.

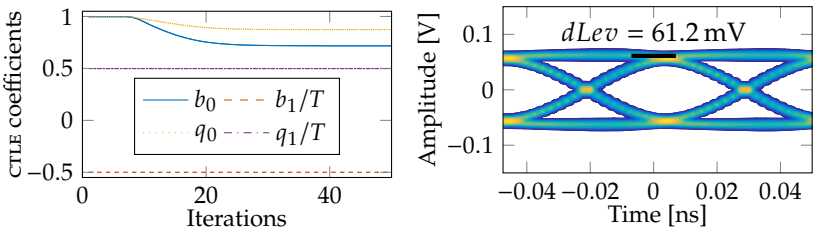


Figure 3.9: Adaptive second-order CTLE applied to the channel in Figure 3.4: (a) Convergence of the CTLE taps to $b_0 = 0.7191$, $b_1/T = -0.5$, $q_0 = 0.8751$, $q_1/T = 0.5$ (with $\mu_{\text{CTLE}} = 0.01$) (b) equalized eye diagram with $dLev = 61.2\,\text{mV}$ (with $\mu_{dLev} = 0.01\,\text{V}$). The second order CTLE has coefficients $c_0 = 0.6292$, $c_1/T = -0.078$ and $c_2/T^2 = -0.25$. The CTLE poles were assumed to be $w_{\text{p1}} = 20\,\text{GHz}$ and $w_{\text{p2}} = 30\,\text{GHz}$ and have been added to the channel transfer function.

Notice that this procedure tends to cancel the first post-cursor and for this channel the SNR is not significantly improved due to the large pre-cursor, as can be seen from the eye in Figure 3.8b.

As an example of adaptation of a second-order CTLE, Figure 3.9 considers the case with a negative and a positive zero (see the values of b_1 and d_1 in Figure 3.9a) that is equivalent to an FIR filter with one pre-, the main and one post-cursor. The eye diagram (Figure 3.9b) is significantly improved by the cancellation of the first pre- and post-cursors.

3.3.1.6 Optimal Sampling Phase

The caveat for the sampling phase SS-LMS loop to work is that it is necessary that some amount of pre-equalization be applied to the signal, so that the main cursor's amplitude (and hence the SNR) is not compromised during the adaptation process. This condition can be accomplished by simply applying FFE at the transmitter,

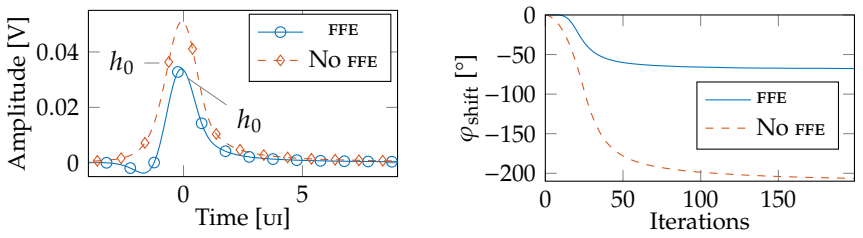


Figure 3.10: Sampling phase adaptation applied to the channel in Figure 3.4 without (dashed red line) and with (solid blue line) FFE featuring $w_{\text{-}1} = -0.25\,\text{V}$ and $w_0 = 0.75\,\text{V}$: (a) pulse responses; (b) evolution of the sampling phase φ_{shift}.

without the need to adapt it, since the final adjustment of the pre-cursor will be handled by the optimal sampling point algorithm. In such a way, the FFE can work with specified pre-sets instead of fully-adaptive LMS algorithms that would require a back-channel [78].

An example of this is presented in Figure 3.10 without CTLE or DFE: plot (a) shows the pulse response without (dashed red line) and with (solid blue line) an FFE featuring $w_{\text{-}1} = -0.25\,\text{V}$ and $w_0 = 0.75\,\text{V}$, highlighting the cursors after sampling. Note that in the case with FFE the main cursor is reduced insomuch as the first pre-cursor reaches 0 V (thanks to the pre-emphasis), whereas without FFE the main cursor is sampled so early that the first post-cursor even becomes larger;[9] moreover, as can be seen from Figure 3.10b, the sampling phase adaptation without FFE would cause the slicers to sample the received signal for data even earlier than the CDR does for edges (180° away from the point of maximum amplitude of the pulse response): This makes no sense and would cause the CDR to fail.

3.3.1.7 Putting it All Together

As an example of the power of such a modelling approach, in [60] the fully-adaptive equalization of an HSSI transmitting at 20 Gb/s on a channel attenuating 12 dB at 10 GHz was demonstrated. The LC π-network describing the package was included in the model of the channel and a fixed 2-tap FFE with a pre-emphasis of approximately 6 dB ($w_{\text{-}1} = -0.25$, $w_0 = 0.75$) was applied at transmit side. Figure 3.11 shows the resulting simultaneous adaptation of $dLev$, CTLE, DFE and sampling phase as a function of the number of iterations performed by the fully-adaptive algorithm described above; Figure 3.12 shows Fourier transforms of relevant signals and blocks in the simulator, the unequalized eye diagram, the channel pulse response prior to equalization, after the fixed FFE and after full adaptation, a situation to which corresponds the last eye diagram.

3.3.2 Behavioural Implementation of the SS-LMS Algorithm

As previously mentioned, a behavioural Verilog-A version of the MATLAB code was developed in order to test the fully-adaptive algorithm directly on a post-layout

[9] Although post-cursors can be easily eliminated with DFE, implementing taps to correct post-cursors that are larger than the main would be impractical.

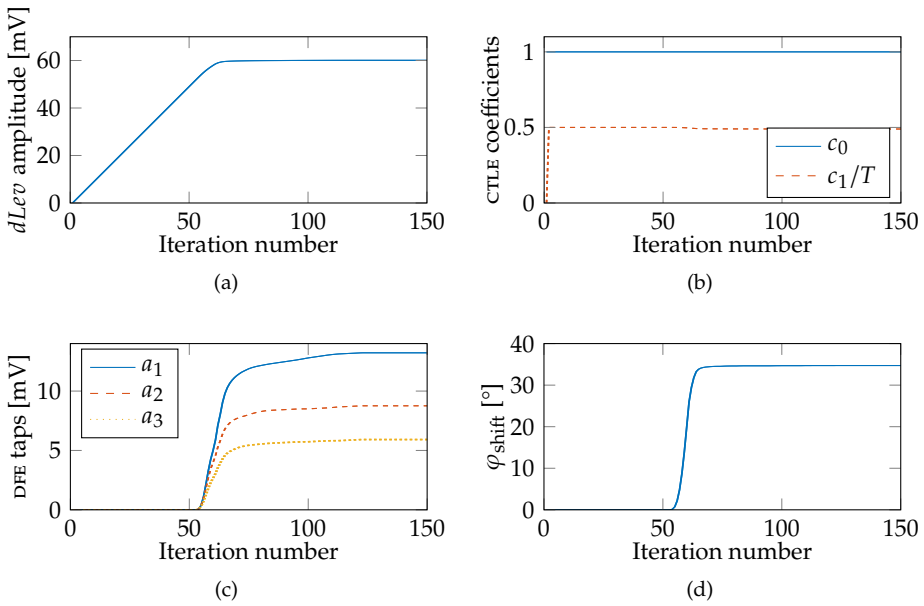


Figure 3.11: Simulation of a channel with 12 dB attenuation at 10 GHz and fixed 2-tap FFE providing ≈ 6 dB of pre-emphasis. (a) *dLev* drives adaptation of (b) first-order CTLE, (c) 3-tap DFE and (d) optimal sampling point.

transistor-level implementation of Infineon's HSSI, as described in [76]. Such a work marked as well an important step towards the final implementation of the SS-LMS algorithm, as the features of the analog circuitry had to be taken into account when considering the characteristics of the adaptive machinery's inputs and outputs, as well as its internal functioning: Mentioning just a few examples, the allowed steps for the optimal sampling point adjustments were dictated by the PI granularity, and the DFE's least-significant bit (LSB) amplitude was dictated by the corresponding circuit implementation. While this is rather obvious, such a step was nevertheless relevant to test the robustness of the work done so far, as real circuits' constraints started to apply.

It is worth to recall that the MATLAB version of the code is *time-agnostic*, i.e. it assumes that a sufficient amount of time has passed between any two consecutive updates so that it has received every possible sequence made up of a certain number of bits (such a number is usually determined based on the number of relevant cursors of the channel pulse response). Needless to say, this is not the case for a real application, where the algorithm itself is clocked and takes as input a finite number of bits (therefore, a subset of all the possible sequences) before updating the equalizer's coefficients. Moreover, working only with a limited number of deserialized data makes it necessary to drop some of the correlations expressed by Equation 3.31, due to the fact that some of the required data bits would be in the preceding or subsequent vector. Storing additional past data bits and/or waiting for the upcoming ones would be required in order to cope with this, but the relative hardware and designed overhead would not be justified: As the SS-LMS technique is based on a probabilistic approach, it is not important which data are considered for the correlations and the corresponding averages as long as a sufficient amount

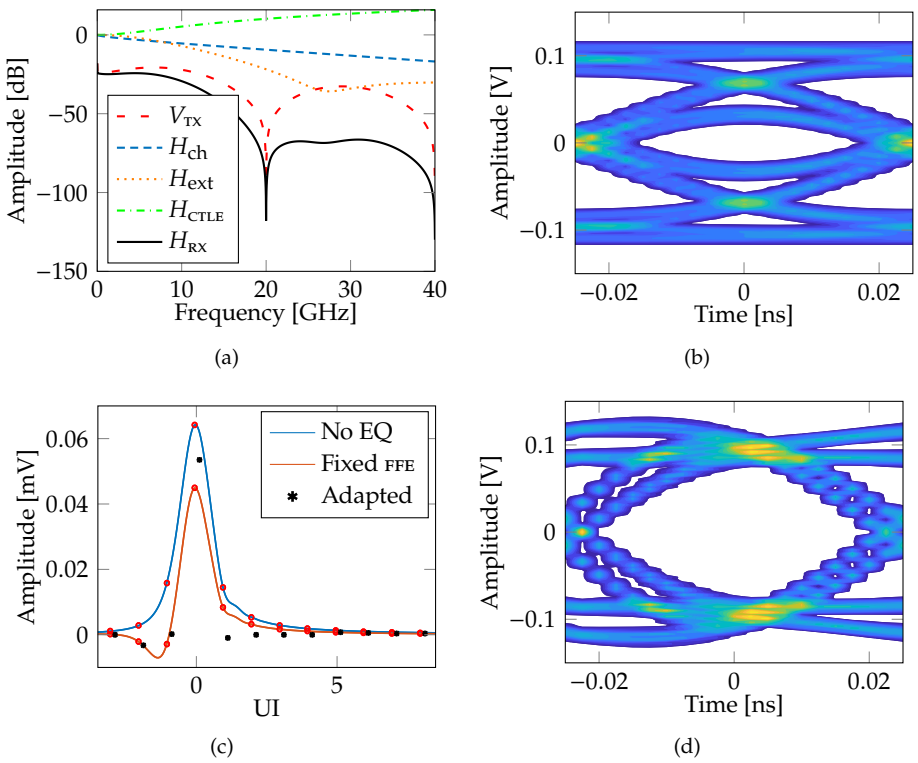


Figure 3.12: Simulation of a channel with 12 dB attenuation at 10 GHz and fixed 2-tap FFE. All the relevant Fourier transforms of signals and transfer functions of the transceiver are plotted in (a); the eye diagram prior to equalization is shown in (b); (c) shows the pulse response prior to equalization (solid blue line), after the fixed FFE (solid orange line) and after application of the fully-adaptive loop (Figure 3.11) and subsequent sampling (black asterisks); (d) shows the resulting equalized eye diagram

of samples is collected.

Once the aforementioned points were addressed, the behavioural model was introduced in the analog circuit developed by Infineon and described in [54,55,109], as well as in Section 4.1.1. The analog circuit was slightly modified in order to insert the error samplers (the ones which compare $dLev$ and y_i in Figure 3.1, implemented only behaviourally at the time) and a behavioural *virtual eye monitor* (shown in Figure 3.13), which was devised in order to observe an effective eye diagram in simulation, regardless of the speculative nature of the DFE and the half-rate architecture of the receiver: The analog signals $y_{e/o,p/n}$ at the output of the DFE (i.e. at the slicer's input) on all four branches of the speculative DFE in the half-rate receiver are delayed by 1 UI to match the sampler's delay; then, speculation is resolved in each (even/odd) half of the receiver by behavioural analog multiplexers driven by the sliced data $\hat{d}_{e/o,p/n}$ coming from the other (odd/even) half receiver (the same procedure, although not shown, is performed to resolve speculation on the data, see Figure 4.1); eventually, the multiplexers' outputs $y_{e/o}$ are further multiplexed through clk_{90} (as selection signal) to combine the equalized analog

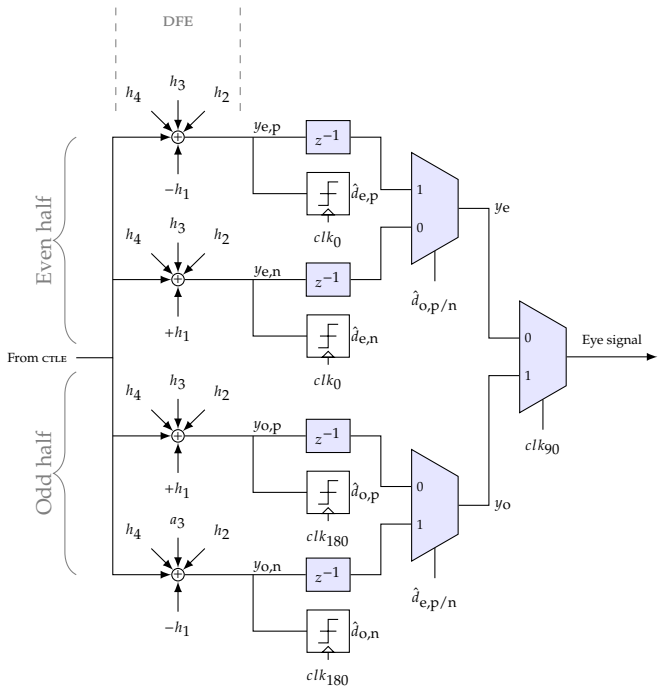


Figure 3.13: Schematic of the virtual eye monitor used in simulations to visualize the effect of the full DFE, despite the first tap being speculative. All shaded blocks are behavioural models of analog circuits; a signal $x_{a,b}$ belongs to the even/odd half receiver for $a = e/o$, respectively, and to the speculative DFE path that subtracts/adds the first tap for $b = p/n$, respectively.

signals into the *eye signal* according to the half-rate functioning of the receiver, hence allowing the eye diagram (that is a sort of "virtual" eye, not measurable at any point of physical system) to be visualised during simulations.

The channel model described in Section 2.2.3 was brought into the analog simulator by fitting the channel's frequency response to the rational function

$$H(s) = \left(\sum_{l=1}^{M} \frac{B_l}{s - A_l} + C \right) e^{-sD}, \tag{3.38}$$

and by implementing it in a Verilog-A block in the form of an analog filter, which can be instantiated in typical SPICE-like circuit simulators [21]. The transfer function resulting from such a fit is shown by the red dashed line in Figure 3.14a and diverges from the analytical one above approximately 12 GHz; however, the resulting pulse response shown in Figure 3.14b provided by the Verilog-A block does not differ significantly from the one produced by inverse Fourier transform of the complete transfer function convolved with the transmitted pulse using the method presented in [74], hence validating such an approach.

The channel features many post-cursors and two large pre-cursors, the peak is significantly attenuated to approximately 50 mV (while the transmitted differential swing was 0.6 V) and the corresponding simulated eye diagram (not shown) is completely closed and corresponds to a very high BER. In such a situation, DFE

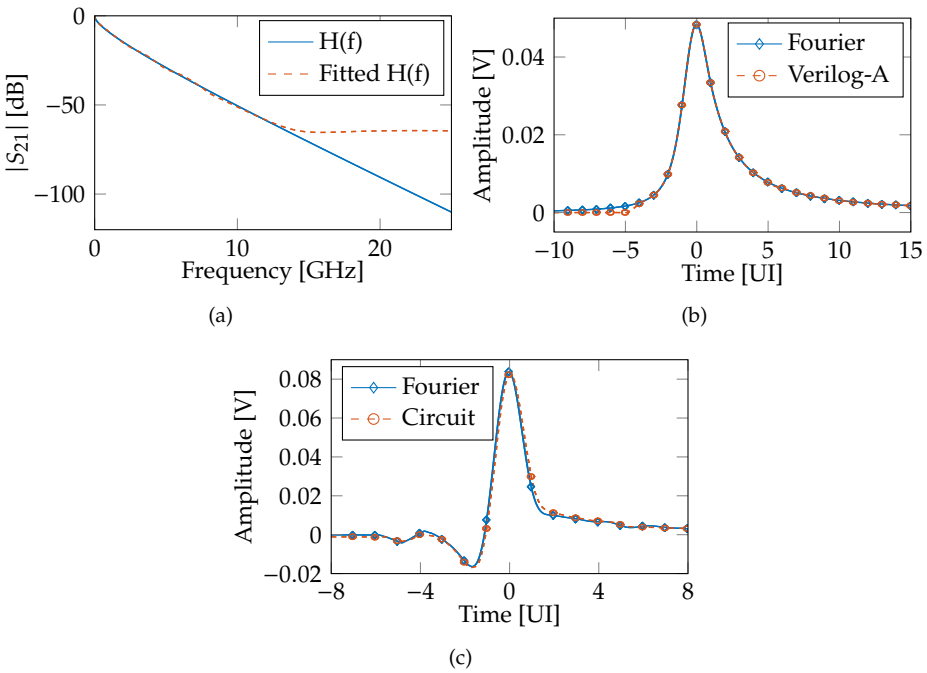


Figure 3.14: (a) Insertion loss and (b) pulse responses of a channel targeted by MIPI A-PHY applications having 33 dB loss at 6 GHz. The solid line is obtained with the full microstrip model presented in Section 2.2.3, the dashed line is the response to its rational fit of Equation 3.38. All equalizers and amplifiers are turned off. (c) shows the pulse response with an FFE pre-set of $w_{-1} = -4/16$, $w_0 = 9/16$, $w_1 = -3/16$, the CTLE peaking 1 dB at Nyquist frequency with respect to the 4 dB low-frequency gain and the two VGAs both amplifying 5.5 dB at Nyquist frequency. The solid line is the inverse transform of the transfer function in Equation 3.38 convolved with the transmitted pulse, while the dashed line is plotted from transistor-level simulations. The amplifiers attempt to counteract the peak reduction due to the strong FFE, resulting in only a slight decrease of the main cursor with respect to plot (b).

alone cannot compensate ISI since its feedback register always contains wrong bit decisions. Therefore, the system's pre-set (VGAs, CTLE and FFE) have to be adjusted to improve the reception. Those need not be optimal settings, and may be determined in a calibration procedure prior to the beginning of transmission. In fact, it will be shown later on that the same pre-settings can be suited also as starting point for different channels, thus demonstrating that they do not need to be chosen precisely. The resulting pulse response is reported in Figure 3.14c: The cursors are greatly reduced, although the eye diagram is still closed, despite the action of the analog front-end (amplifying ≈ 15 dB at Nyquist frequency).

A post-layout transistor-level version[10] of Infineon's HSSI operating at 12 Gb/s

[10]The whole SerDes is modelled at transistor level except for: The four variable-threshold comparators used as error samplers for the fully-adaptive equalization algorithm, the SS-LMS loop itself, the circuitry for the eye monitor shown in Figure 3.13 and the channel, which are all Verilog-A behavioural blocks.

(refer to Section 4.1.1 for a detailed presentation of the transceiver) was then used to test the behavioural SS-LMS algorithm over a channel representing typical specifications of yet-to-be-defined MIPI A-PHY standard, likely to require transmission over long-reach channels attenuating 33 dB at Nyquist frequency [110, 111]. The HSSI in such a configuration was tested with a fixed FFE having $w_{-1} = -4/16$, $w_0 = 9/16$, $w_1 = 3/16$, all 12 driver slices connected to enhance the transmitted pulse's amplitude (although at the cost of TX impedance mismatch, see Section 1.3.2), the CTLE and the two VGAS amplifying respectively 4 dB and 5.5 dB at Nyquist frequency (reaching an amplification of about 15 dB overall), but still resulting in a completely-closed eye diagram. This is shown in Figures 3.15a and 3.15b for eye diagrams computed with the probabilistic approach of Chapter 2 and with the virtual eye monitor presented above, respectively, thus demonstrating a good correspondence between the two methods.

Figure 3.15 also shows the results obtained by activating the fully-adaptive algorithm on a 3-tap DFE and on the sampling phase. Figure 3.15c depicts the behaviour of the SS-LMS behavioural algorithm: $dLev$ converges in the proximity of 80 mV; the optimal sampling point moves to about 70° away[11] from the edge clock phase (compared to the 90° ideally set by the CDR) in order to reduce the first pre-cursor; the three DFE taps converge to 70 mV, 14 mV and 14 mV respectively, rather similar to the post-cursors' amplitude in the corresponding pulse response after the sampling point has moved. In Figures 3.15d and 3.15e the resulting eye diagram is shown for the two simulation methods, with significant vertical and horizontal apertures proving the effectiveness of the applied equalization, also supported by the null BER in the circuit simulation after a sequence of 10^4 bits; the visible asymmetry in the eye is due to the change in sampling point after adaptation, which directs sampling towards a point with minimal ISI and modifies the point of application of the DFE as a consequence.

Although the time-domain, transistor-level simulations cannot prove functioning with BER $< 10^{-12}$ (as required by standards such as MIPI A-PHY and IEEE 802.3), the post-processing of the pulse response suggests that this is the case, provided that the jitter of the clock and the CDR is sufficiently low. To have a rough estimate of such an effect, the bathtub plot measured at 4.6 Gb/s in [55] for the previous version of the chip was fitted by assuming jitter characterised by a Gaussian distribution [112] as shown in Figure 3.16a, yielding an RMS jitter of about 4 ps. Assuming that the clock synthesiser for the 12 Gb/s SerDes is of quality similar to that running at 4.6 GHz in [55], the product *frequency* × *jitter* should remain the same (see Equation A.6), so that the 12 GHz clock considered in this Section may be affected by an RMS random jitter of approximately 1.5 ps; the result on the bathtub in Figure 3.16b, simulated with the approach of Sections 2.2.5 and 2.2.6, is a reduction in eye width from ≈ 38 ps to ≈ 19 ps at BER $= 10^{-12}$.

[11] Such a value is actually extracted from the simulated waveforms by evaluating the distance between the sampling clock's edges and the signal's transitions, not by simply looking at the output of the SS-LMS loop. The transistor-level simulations were run without a CDR to reduce the runtime, hence the starting sampling phase depends on the simulation settings, and it may be far off the desired point: Adaptation of φ_{shift} in Figure 3.15c starts from this value, which does not have an exact correspondence with a precise position in the eye diagram.

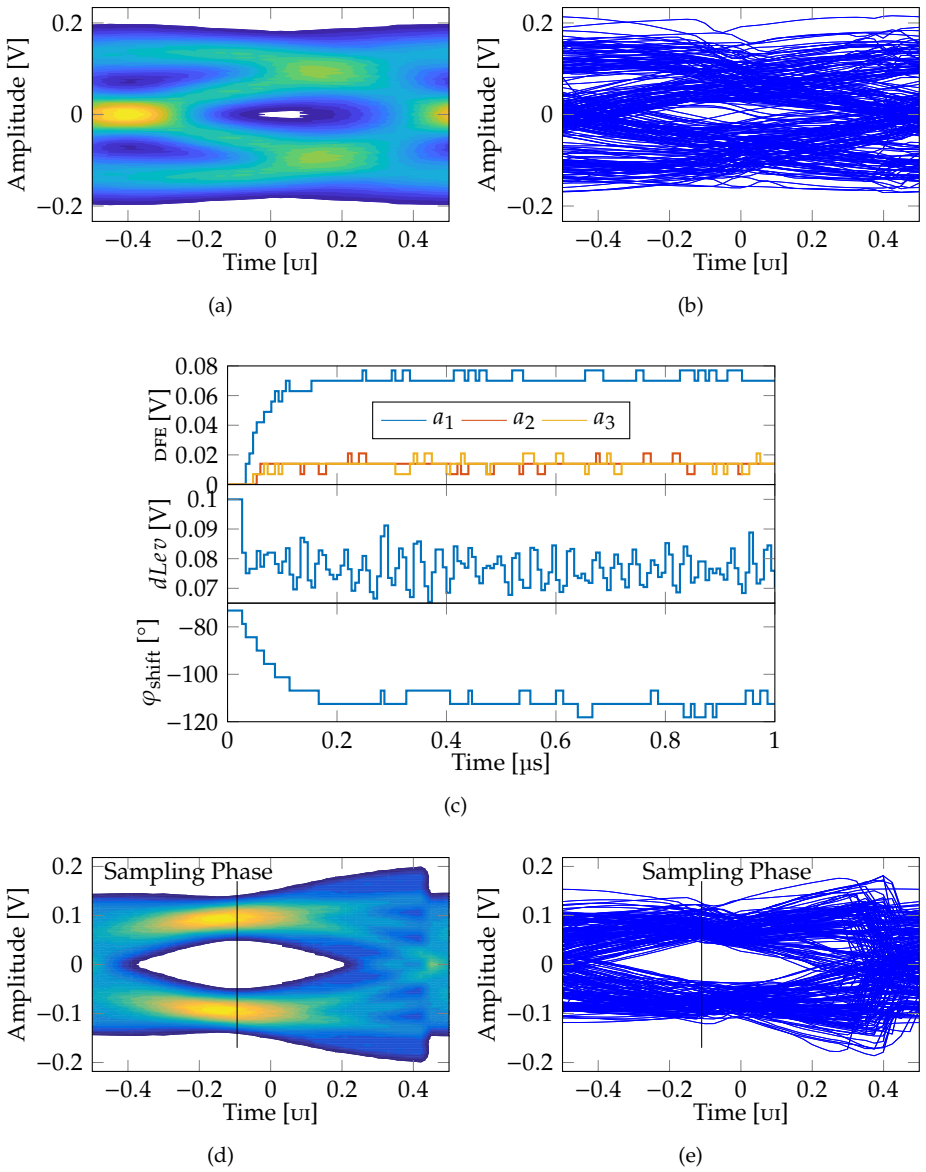


Figure 3.15: Eye diagrams of the MIPI A-PHY channel shown in Figure 3.14 before adaptive equalization (FFE featured $w_{-1} = -4/16$, $w_0 = 9/16$, $w_1 = 3/16$, all 12 driver slices were connected and the AFE provided about an amplification of 15 dB at the Nyquist frequency): (a) computed statistically (see Chapter 2) and (b) obtained through the virtual eye monitor (see Figure 3.13) added to the post-layout transistor-level simulations. The equalization quantities adapted by the behavioural block as a function of time (the DFE taps and the phase are computed as digital codes and were converted to physical quantities for plotting purposes) are shown in plot (c), whereas the eye diagrams after convergence of the SS-LMS equalization loops are shown in plot (d) and (e) for the probabilistic approach and the circuit simulator.

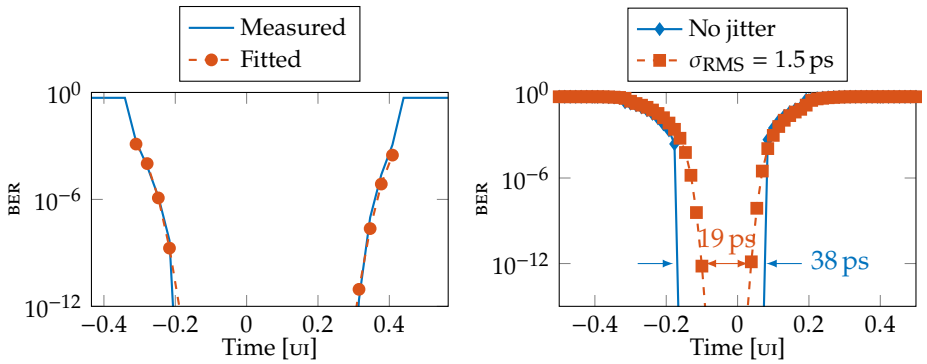


Figure 3.16: Inclusion of jitter into the model for eye diagrams: (a) shows the bathtub measured in [55] and the fit used to extract its jitter ($\sigma_{RMS} \approx 4$ ps), while (b) shows the bathtub corresponding to Figure 3.15d without and with the addition of jitter characterised by $\sigma_{RMS} = 1.5$ ps.

Finally, it will be shown that the FFE, CTLE and VGAS pre-sets chosen above can support fully-adaptive operations for different channels. The same geometry and physical characteristics that resulted in the transfer function of Figure 3.14a were chosen, although now to obtain the model of a channel attenuating 20 dB at Nyquist frequency. The equalized eye diagram obtained through post-layout, transistor-level simulations with the virtual eye monitor is shown in Figure 3.17, demonstrating equalization capabilities over different channels. Differently from Figures 3.15d and 3.15e, now the optimal sampling point is delayed with respect to the centre of the eye, because the chosen FFE pre-set results in a negative pre-cursor in this case, which is compensated by moving the sampling point to the right (in the previous case it was moved to the left).

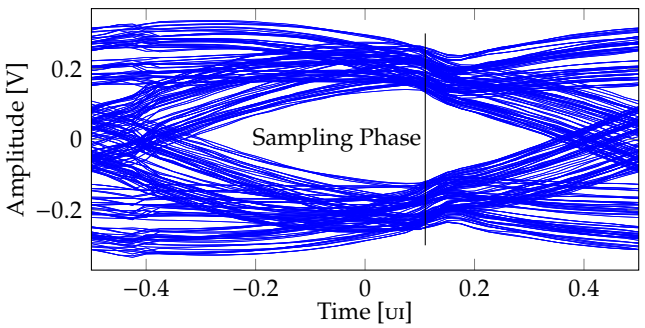


Figure 3.17: Eye diagram after convergence of the behavioural SS-LMS equalization loops on a channel with 20 dB loss at 6 GHz as obtained from post-layout, transistor-level simulation (FFE featured $w_{-1} = -4/16$, $w_0 = 9/16$, $w_1 = 3/16$, all 12 driver slices were connected and the AFE provided about an amplification of 15 dB at the Nyquist frequency).

Chapter 4

Implementation of an Integrated Fully-Adaptive Equalization System

This Chapter focusses on the fully-adaptive equalization system described in Chapter 3 that was implemented on a test chip for this PhD work. However, before designing the digital fully-adaptive algorithm, an experimental assessment of the automotive wireline transceiver previously developed at Infineon (referred to as *test chip 18* or *TC18*) was necessary to understand the performance of the device that the new feature was planned to work on and possible hardware improvements required to improve its operation.

4.1 Laboratory Characterisation of a Previous Chip

4.1.1 Transceiver Architecture and Simulation Results

Figure 4.1 shows schematically the transceiver's architecture for automotive microcontroller applications, omitting the CDR algorithm that is detailed in [109]. It was designed mainly by Andrea Bandiziol from Infineon Technologies in 28 nm planar CMOS technology (characterization over voltage variations was reported in [55], while simulations considering also process variations are shown below). The original design was presented in [55], and modifications were subsequently implemented to increase the transmission speed to 13 Gb/s.

The full-rate transmitter [113] performs FFE with one pre-, one main- and six post-cursors. The output voltage swing is set by a Low-Dropout Regulator (LDO) and the transmitter's output impedance can be adjusted by connecting in parallel up to 12 different replicas of the driver. Each driver replica is made up of 16 slices connected to the FFE taps, which are manually programmable in steps of 1/16 ($\approx$ 1.16 dB of variation, as per Equation 1.4) through a JTAG connection.

The receiver [109] is half rate, meaning that it is clocked at a frequency of $f_B/2$ (i.e. corresponding to half the data rate) and employs both the clock's edges to sample the data; this is achieved by replicating the receiver circuit in two identical even and odd (only the even half is shown in Figure 4.1) parts being timed by two clocks with opposite phase (i.e. clk_0 and clk_{180}, with the terminology of Figure 4.1).

The DFE features one speculative tap, hence requiring two summing stages per path (for a total of four): One that always adds $+a_1$ as the first tap and slices the data $\hat{d}_{e,p}$ when clk_0 rises (for the even half shown in Figure 4.1), the other one that applies $-a_1$ and samples $\hat{d}_{e,n}$ instead. The decision concerning which of the

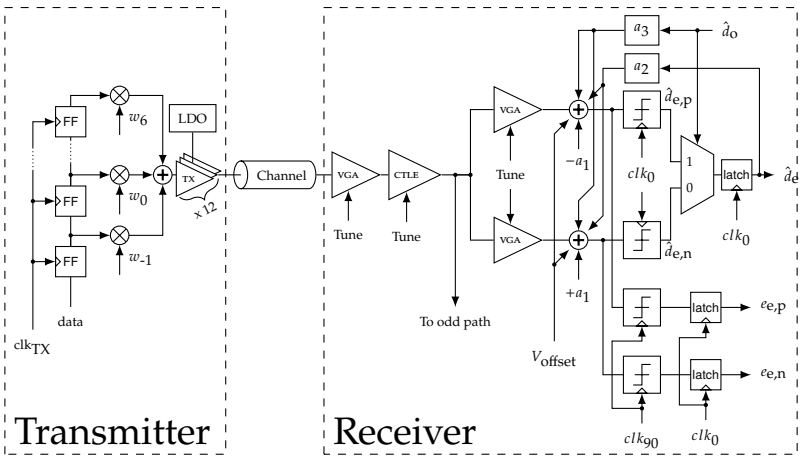


Figure 4.1: Architecture of the transceiver (differential lines are drawn as single-ended). On the left, the TX is shown with FFE; the transmission channel is followed by an AFE comprising tunable VGA and CTLE; the subsequent even portion of the half-rate RX (the odd portion is not shown) implements a 3-tap speculative DFE with samplers featuring manual offset compensation; edge sampling is shown on the bottom right. The CDR (not shown) provides the edge and the data clocks. A signal $x_{\mathrm{a,b}}$ belongs to the even/odd half receiver for $a = e/o$, respectively, and to the speculative DFE path that subtracts/adds the first tap for $b = p/n$, respectively

two data samples to retain is taken through a multiplexer driven by the data $\hat{d}_{\mathrm{o}}$ sampled at the previous sampling instant (of clk_{180}) and already "speculated".

At the receiver's input a programmable *variable-gain amplifier (VGA)* sets the voltage swing for the following CTLE. Another programmable VGA is embedded in each of the four DFE summing nodes (considering the half-rate, loop-unrolled architecture). The CDR produces for each receiver's half stage half-rate edge and data clocks ($clk_{0/180}$ and $clk_{90/270}$, respectively) with 90° spacing employing a PI with 64 steps; due to the half-rate architecture, a UI is covered by 32 steps only.

Compared to the implementation reported in [55], various blocks were re-designed for TC18 to support a higher speed: The VGAs in such an implementation have a gain that spans from −6 dB to 6 dB in order to set the most appropriate voltage swing for the subsequent blocks; the CTLE can compensate up to 7.5 dB; one of the DFE taps has been devoted to offset-compensation of the samplers at start-up time (V_{offset} in Figure 4.1); other components were simply optimized for such an increased speed (e.g. the bandwidth of the PI was increased). Robustness over PVT corners was ensured by trading off speed, area and power: Minimum-area devices were avoided and safety margins under nominal conditions were kept.

Following the aforementioned upgrade, the transceiver's figure of merit reported in [55] has increased to 6.8 mW/(Gb/s) w.r.t. 5.7 mW/(Gb/s) in Table I therein. Post-layout transistor-level simulations including parasitics (see Table 4.1 here) demonstrate operation of the transmitter at BER below 10^{-12} up to 13 Gb/s at room temperature with nominal conditions, while at 12 Gb/s the whole automotive range of temperature (−40 °C to 150 °C), supply (−10 % to +10 %, demonstrated experimentally in [109]) and technology corners (only slow-slow and fast-fast results are reported for brevity) can be covered. The whole transceiver was simulated

in loopback mode and no errors were found in the received $\sim 2^{15}$ bits over the whole automotive PVT range. Longer mixed-signal simulations on *XA-VCS* also showed no errors.

Corners			Figures of merit	
Voltage [%]	Temperature [°C]	Technology	EH [mV]	EW [ps]
10	150	FF	596	75
10	150	SS	487	74
10	−40	FF	626	76
10	−40	SS	505	81
−10	150	FF	556	76
−10	150	SS	436	73
−10	−40	FF	549	75
−10	−40	SS	438	73

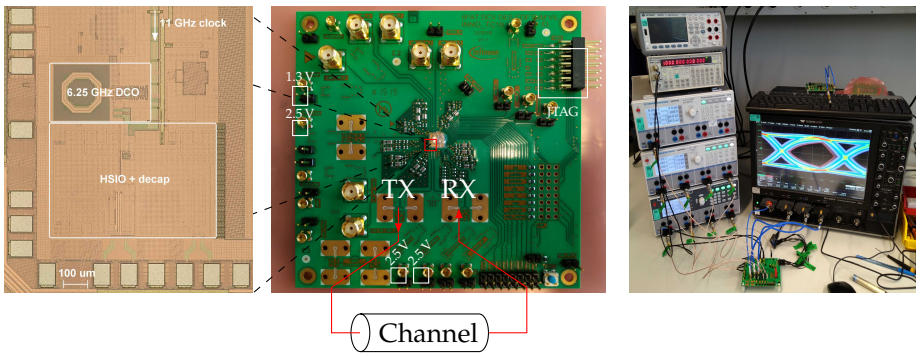
Table 4.1: Simulated eye height (EH) and width (EW) of TC18 for automotive corners at 12 Gb/s. Supply parasitics: R = 1 Ω and L = 1 nH; ground parasitics: R = 200 mΩ and L = 200 pH.

4.1.2 Experimental Characterisation

The following is the result of the characterization performed at Infineon Technologies Villach (Austria), mainly by Dylan D'Ampolo from University of Udine and Infineon Technologies with the measurement setup depicted in Figure 4.2. It consists of the main PCB on which the transceiver is mounted, one or more auxiliary PCBs with several test channels, power supplies and a Teledyne Lecroy SDA 830Zi-B 30 GHz Serial Data Analyzer; temperature measurements were done by means of an S-tec TSR 2252-LN thermostream (not shown). An FTDI module connected to a PC is used to program the chip. The chip is contained in a thermally-proven Quad Flat No-Lead package characterised by per-pin 4.5 nH inductance (including bondwire), 400 nF capacitance and 400 mΩ resistance.

By means of a VNA the S parameters of all channels of the auxiliary PCB were measured over a 20 GHz range: The 60 cm microstrip and the 10 cm stripline used in the tests lose respectively 10 dB and 12.5 dB at the corresponding Nyquist frequencies of 3.125 GHz and 5.5 GHz. An issue related to such channels is given by their S_{11} that is rather high at Nyquist frequency (≈ -2.5 dB and -10 dB, respectively), due to connectors trace mismatch in the host PCB (single-ended impedance of the traces is 75 Ω); this entails that, even though their attenuation *per se* is not very high, the mentioned reflecting behaviour degrades the quality of the transmission. Measured pulse responses of the two channels corresponding to the respective frequencies of interest are reported in Figure 4.3a.

The eye diagram at the output of the transmitter at the top speed of 13 Gb/s with *pseudo-random bit sequence (PRBS)*31 is shown in Figure 4.3b. The board was connected directly to the scope via 30 cm cables, with estimated attenuation at the Nyquist frequency of approximately 2 dB. However, at this rate, operation at BER below 10^{-12} over the whole automotive temperature (−40 °C to 125 °C) and voltage range (−10 % to 10 %) is not guaranteed, thus lower rates of 11 Gb/s and 6.25 Gb/s are selected in the following. Figure 4.4 shows the eye diagrams at 6.25 Gb/s over



(a) (b)

Figure 4.2: Figure (a) shows the micrograph of TC18 and the PCB used for testing, highlighting the voltage supplies, the JTAG connector and the TX and RX channels. Figure (b) shows the measurement setup connected to a 60 cm-long microstrip channel.

the 60 cm channel. Without FFE (4.4a) the eye is closed but compensation of the first pre-and first post-cursors is effective in opening the eye (4.4b).

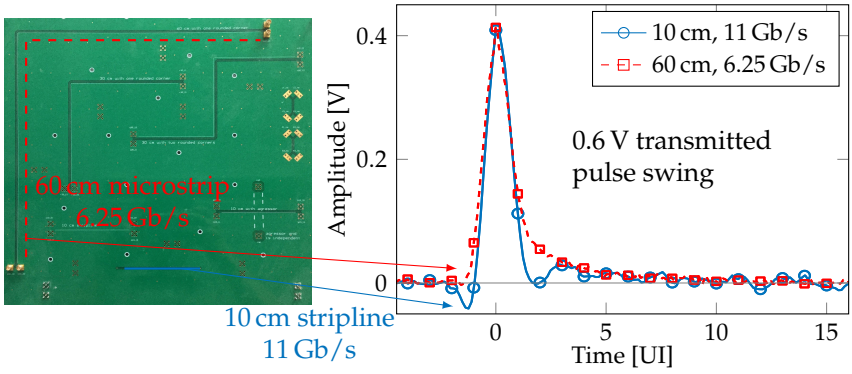
Figure 4.5a shows the bathtub at 6.25 Gb/s with the main PCB and 30 cm-long connection cables only (for an overall 2 dB loss), Figure 4.5b with an additional 60 cm-long microstrip channel adding 10 dB loss at Nyquist frequency. Notice that without the 60 cm microstrip there is a large timing interval with BER $< 10^{-12}$ and this has a minor dependence on temperature. The presence of the 60 cm microstrip reduces the bathtub opening, especially at high temperature.

Figure 4.6 shows the bathtub at 11 Gb/s: As in the 6.25 Gb/s case, when considering only the main PCB and cables (plot a) the bathtub opening is almost independent of temperature and features a significant interval with BER $< 10^{-12}$. The situation worsens when the 10 cm stripline is included (plot b): In this case at 165 °C BER $< 10^{-9}$ cannot be achieved. However, BER $< 10^{-12}$ is possible at 125 °C showing that the circuit is able to cover the automotive range even with a channel losing 14.5 dB. In this respect, Table 4.2 summarizes relevant metrics for the eye diagrams and the bathtubs measured at 11 Gb/s without a channel over the automotive temperature corners.

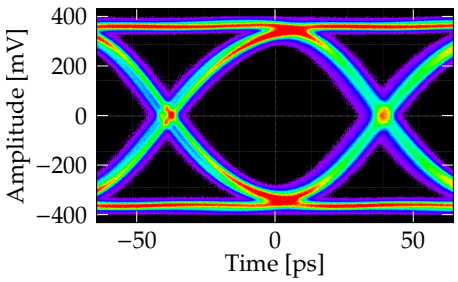
The presence of different supply domains allows one to identify the power drawn by different blocks, and these are reported in Table 4.3.[1] The main responsible for power consumption is the RX analog part and the second largest contribution is given by the TX, especially when operating at high bit rates, with the driving circuitry and the preceding serializer, FFE and resampling drawing almost an equal amount of power at both data rates considered here.

A comparison with the (not automotive) state of the art is reported in Table 4.4. Due to large reflections in the auxiliary PCB, the system was tested with channels attenuating up to 14.5 dB. Notice that the simulations with impedance-matched microstrips reported in Section 4.1.1 showed that TC18's FFE and DFE can equalize losses as high as 33 dB.

[1]Notice that, due to an error in the evaluation of power consumption, the transmitter power differs from what was reported in [114].

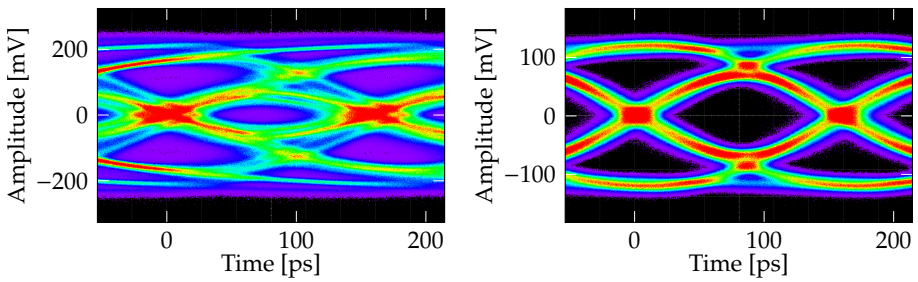


(a)



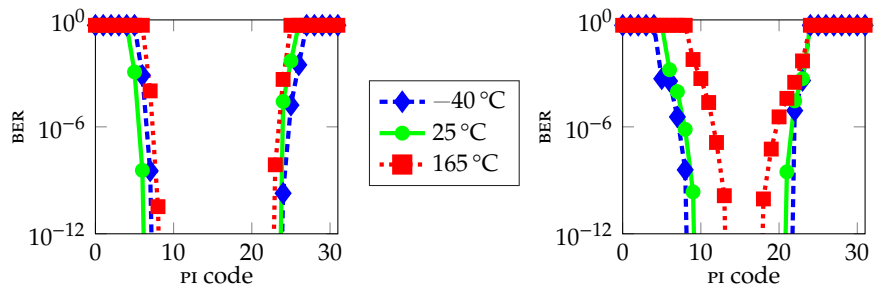
(b)

Figure 4.3: Figure (a) shows the PCB channel board and the measured pulse responses for a 60 cm channel at 6.25 Gb/s (solid blue line) and for a 10 cm channel at 11 Gb/s (dashed red line), highlighting the corresponding PCB trace. A single acquisition was set through the oscilloscope for sequences containing a single '1' bit. Figure (b) shows the measured eye diagram at the output of the transmitter at 13 Gb/s without FFE; 12 driver replicas are activated, the supply is $V_{DD} = 0.9$ V.



(a) (b)

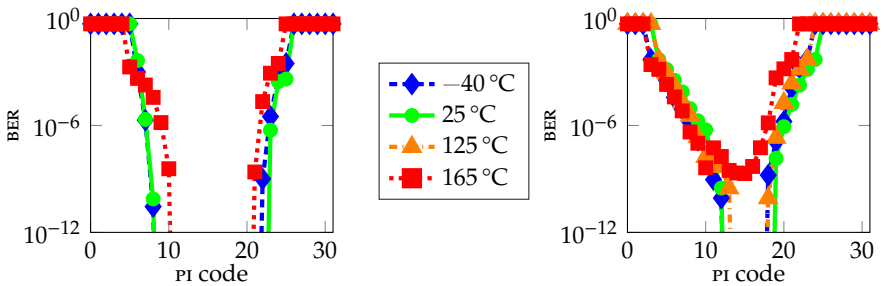
Figure 4.4: Eye diagrams at the transmitter at 6.25 Gb/s with a 60 cm-long microstrip channel losing 10 dB at Nyquist frequency. The FFE is disabled in (a), while in (b) has weights $w_{-1} = -1/16$, $w_0 = 12/16$ and $w_1 = -3/16$; 10 driver replicas are activated and the supply voltage is $V_{DD} = 0.9$ V.



(a) FFE ($w_{-1} = -1/16$, $w_0 = 14/16$, $w_1 = -1/16$), PRBS31, DFE ($a_1 = 28\,\text{mV}$, $a_2 = 21\,\text{mV}$)

(b) FFE ($w_0 = 15/16$, $w_1 = -1/16$), PRBS31, DFE ($a_1 = 28\,\text{mV}$, $a_2 = 21\,\text{mV}$, $a_3 = 7\,\text{mV}$)

Figure 4.5: Bathtub plots measured at different temperatures in loopback mode at 6.25 Gb/s (a) with the main PCB and 30 cm-long cables losing overall 2 dB at Nyquist frequency and (b) with an additional 60 cm-long microstrip channel adding a 10 dB loss at 3.125 GHz. Each PI code corresponds to 5 ps at 6.25 Gb/s. 10 driver replicas are activated and the supply voltage is $V_{DD} = 0.9\,\text{V}$; the VGA and the CTLE amplify respectively 4.5 dB and 5.5 dB at Nyquist frequency.



(a) FFE not activated, PRBS32, 9 driver replicas, DFE ($a_1 = 70\,\text{mV}$, $a_2 = 42\,\text{mV}$, $a_3 = 14\,\text{mV}$)

(b) FFE ($w_0 = 14/16$, $w_1 = -2/16$), PRBS8, 10 driver replicas, DFE ($a_1 = 70\,\text{mV}$, $a_2 = 42\,\text{mV}$, $a_3 = 7\,\text{mV}$)

Figure 4.6: Bathtub plots measured at different temperatures in loopback mode at 11 Gb/s (a) with the main PCB and 30 cm-long cables losing overall 2 dB at Nyquist frequency and (b) with an additional 10 cm-long stripline channel adding a 12.5 dB loss at Nyquist frequency. Each PI code corresponds to 2.8 ps at 11 Gb/s. The supply voltage is $V_{DD} = 0.9\,\text{V}$; the VGA and the CTLE amplify respectively 4.5 dB and 5.5 dB at Nyquist frequency.

Corner	Figures of merit			
	Eye		Aperture [UI] @ BER=	
T [°C]	Height [mV]	Width [ps]	10^{-9}	10^{-12}
−40	471	77.1	0.447	0.428
25	463	77.1	0.472	0.456
150	440	76.7	0.341	0.334

Table 4.2: Measured bathtub apertures, eye height and width for automotive corners at 11 Gb/s. No FFE, 9 driver replicas are activated and V_{DD} = 0.9 V. The DFE has taps a_1 = 70 mV, a_2 = 42 mV and a_3 = 14 mV, the VGA and the CTLE amplify respectively 4.5 dB and 5.5 dB at Nyquist. Main PCB + cables.

	Power Efficiency [mW/(Gb/s)] @	
	6.25 Gb/s	11 Gb/s
Transmitter	3.9	7.0
Driver + Pre-driver	1.9	3.5
Serializer + FFE + Resampling	2.0	3.5
Receiver	4.8	5.9
VGA + CTLE + DFE + Slicer	3.5	4.3
PI + Deserializer	1.3	1.6
Total	8.7	12.9

Table 4.3: Power consumption normalised to the bit rate [mW/(Gb/s)] at 6.25 Gb/s and 11 Gb/s for V_{DD} = 0.9 V. Main PCB + cables.

	This work	[115]	[116]	[117]	[36]	[79]
Node [nm]	28	65	65	45	32	28
Speed [Gb/s]	11	10	10	19	28	20
Channel loss [dB]	14.5	23	35	25	29	20
FFE taps	8	0	0	4	4	2
DFE taps	3	2+1[a]	2 IIR	5	15	2
Power [mW/(Gb/s)]	12.9	2.44	0.99	6.2	18	6.5
V_{DD} [V]	0.9	1	1	1.1	1.1	1.35
Area [mm^2]	0.08[b]	0.25[c]	0.03[c]	0.07[c]	0.81[d]	0.12[e]

[a]Edge DFE tap

Layout areas include: [b]TX+RX, [c]RX, [d]TX+RX+PLL/4, [e]TX+RX+PLL

Table 4.4: Transceiver performance at 11 Gb/s with 10 cm stripline and comparison with similar works.

4.2 New Chip: Hardware Upgrade for Fully-Adaptive Equalization

Following the characterisation of TC18 presented in the Section above and based on knowledge gathered during its design phase, a few critical blocks were deemed to require some modifications either to improve performance or simply to allow the fully-adaptive equalization circuitry to work on this architecture without having to re-design it from scratch. Such an upgraded version shall be named TC20.

4.2.1 Transmitter

The main goal on the transmit side was to reduce power consumption while possibly improving performance at the same target data rate of 12 Gb/s, possibly operating up to 13 Gb/s. This was accomplished by reducing the tap range of the FFE and by optimising the pre-driving stage in order for them to induce less switching noise on the voltage supplies (refer to Appendix A.4 for details on the consequences of noisy supplies).

As shown in Section 4.1.1, the FFE in TC18 was composed of 8 taps, of which 6 were devoted to post-cursors equalization. This can be very effective when trying to equalize all ISI through de- and pre-emphasis by employing the MSE or the SS-LMS approach outlined in Sections 1.3.3 and 3.2.4, respectively, but has many drawbacks in real applications: First of all, the channel's pulse response has to be known to find the FFE weights that minimise the MSE, thus requiring some sort of bidirectional communication or back channel to provide the transmitter with such information (typically acquired at the receiver) [31,78]; secondly, when employing a large number of FFE taps the voltage swing at the transmitter's output is reduced, as was shown in Section 3.3.1, hence making detection more difficult at the receiver. One additional reason for reconsidering the FFE tap range is that post-cursors can be effectively equalised at the receiver by the DFE and, although to a lesser extent, by the CTLE, thus making FFE post-taps relatively redundant.

This has led to the transmitter of Figure 4.7, which uses the same 40:1 serializer and driver of TC18, but features an FFE with three taps instead of eight, a smaller FFE

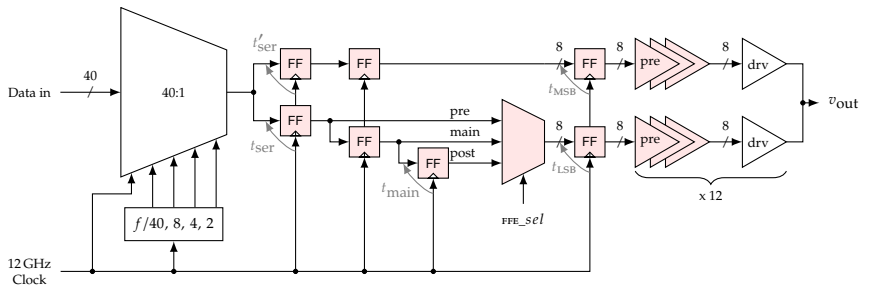


Figure 4.7: Simplified block diagram of the transmitter of TC20: The parallel input data are serialized and fed at full rate to the three-tap FFE, which is applied through 16 programmable driver slices (8 are always connected to the main tap); 12 driver replicas are employed to set the output impedance. The circuits that were upgraded from TC18 (FFE and pre-driver) are shaded; all signals after the serializer are differential.

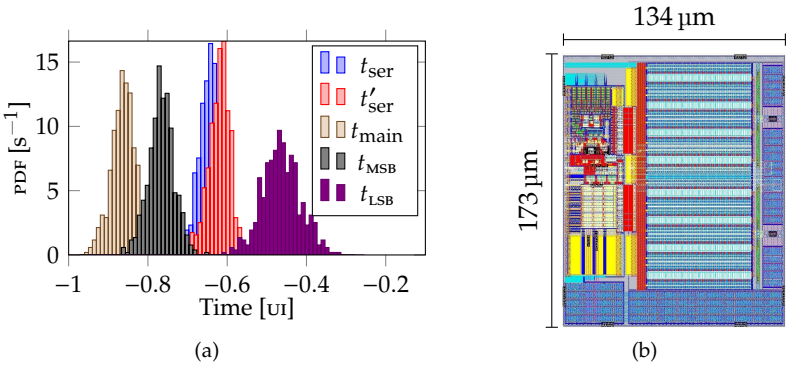


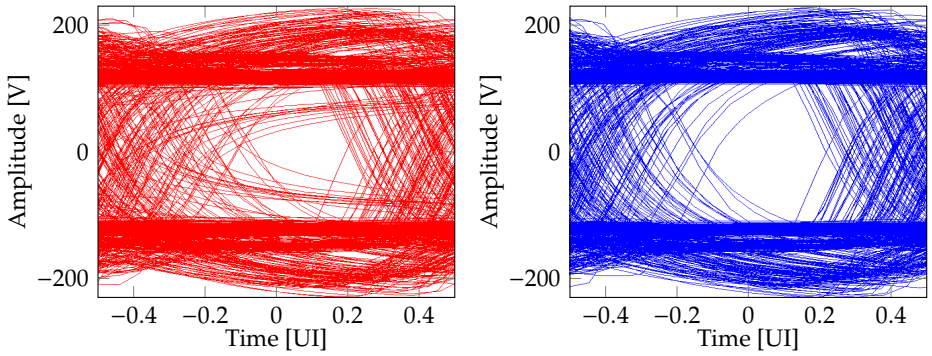
Figure 4.8: Layout of the upgraded transmitter for TC20.

selection circuitry (as a consequence of tap reduction) and optimised pre-drivers. As in TC18, the FFE is set through 16 programmable driver slices, 8 of which are always connected to the main tap, whereas driver's output impedance is set by manually enabling up to 12 replicas of this structure.

While reduction of the tap range was rather straightforward to perform at the circuit level (as it only involved the removal of the extra resampling stages and multiplexers of TC18 [113] and the modification of the local clock tree for the new, reduced load), more effort was devoted to the corresponding layout as the area that became available allowed the optimisation of sensitive interconnections for signal integrity of clock and high-speed data, as well as the placement of decoupling capacitors to try and reduce the amount of switching noise induced into the power supply. Monte Carlo simulations were run at 13 Gb/s and timing margins (difference between the time the data is available and the sampling edge of the clock) for all resampling stages were verifiedto be respected after the modification, as shown in Figure 4.8a for the signals displayed in gray in Figure 4.7: Both t_{ser} and t'_{ser} were checked in order to consider the effect of different paths in the clock tree; t_{LSB} (on the path of the programmable FFE slices) was the only one to feature a margin lower than 0.5 UI, although always higher than 0.365 UI (28 ps), due to the additional logics required for the tap weights selection.

The pre-driving circuit was rather straightforward to optimise as well, since it only required reducing the number of buffering stages between the FFE and the driver, and it was performed with available library cells rather than custom buffers, as they already provided adequate performance (e.g. no bandwidth limitations, reasonable power consumption) at the target 13 Gb/s data rate; extensive simulations (not shown) were performed to asses the strength of the resulting pre-driving circuit on all corners, especially in terms of slope at the driver and signal integrity. The available area was then employed for local decoupling of the high-frequency switching activity of the pre-driver. The driver of TC18 (detailed in [55, 113]) was left untouched and kept for TC20, since it had been designed for the same data rate and following the laboratory characterisation of Section 4.1.2 it was not deemed a limiting factor in reaching the target speed.

The changes presented above resulted in the layout of Figure 4.8b and proved effective in improving the performance of the transmitter: From post-layout simulations at 13 Gb/s (although faster than the 12 Gb/s target, this data rate was used



(a) TC18 (b) TC20

Figure 4.9: Post-layout simulations at 13 Gb/s of the transmitters (a) implemented in TC18 and (b) after FFE tap reduction and optimised pre-driving stage for TC20. The FFE features $w_{-1} = -1/16$, $w_0 = 12/16$ and $w_1 = -3/16$; 8 driver slices are enabled; worst-case scenario PDN with $L = 2\,\text{nH}$, $R = 1\,\Omega$ and no decoupling capacitor; no channel is included.

to find possible limitations in reaching higher throughputs), the eye width was increased by more than 8 % (from 55.9 ps to 60.78 ps) and a small increase in eye height was also observed (about 1 %). When considering an oversized and unrealistic *power delivery network* (PDN) featuring an inductance of 2 nH, a resistance of 1 Ω and no decoupling capacitor as some sort of worst-case scenario, and an FFE[2] with $w_{-1} = -1/16$, $w_0 = 12/16$ and $w_1 = -3/16$, the peak-to-peak voltage noise on V_{DD} was reduced by $\approx 25\,\text{mV}$ (or slightly less than 8 %) and, as shown in Figure 4.9, the eye diagram relevantly improved both in width and in height.

4.2.2 Receiver

The upgrade in the receiver resulted in the block diagram of Figure 4.10 and was mainly motivated by the hardware requirements of the fully-adaptive equalization circuitry, whereas some further modifications were performed to reduce PSIJ (e.g. by improving the decoupling strategy and optimising the buffer chain connecting the deserializers to the digital macro) and improve the quality of the receiver's clock, hence its jitter performance, by reconsidering the clock distribution network. The main additions related to the new features were a chain composed of PI, CML-to-CMOS converter and buffers to provide the clock to the error samplers, two deserializers for the error samples, the error samplers themselves and the synthesised digital circuits that perform adaptation and eye monitoring, while other modifications are presented in the remainder of this subsection.

The signal coming from the channel is driven through a VGA to set the correct voltage swing for the following CTLE, representing the first equalization stage. The signal then enters the actual half-rate portion of the receiver and is split in four paths to feed the 3-tap, 1-tap speculative DFE shown in Figure 4.11. The half-rate architecture implies that two half circuits exist (even and odd half) that work on

[2]Although equalization was not required in absence of a channel, it was enabled to include the modifications in the simulation.

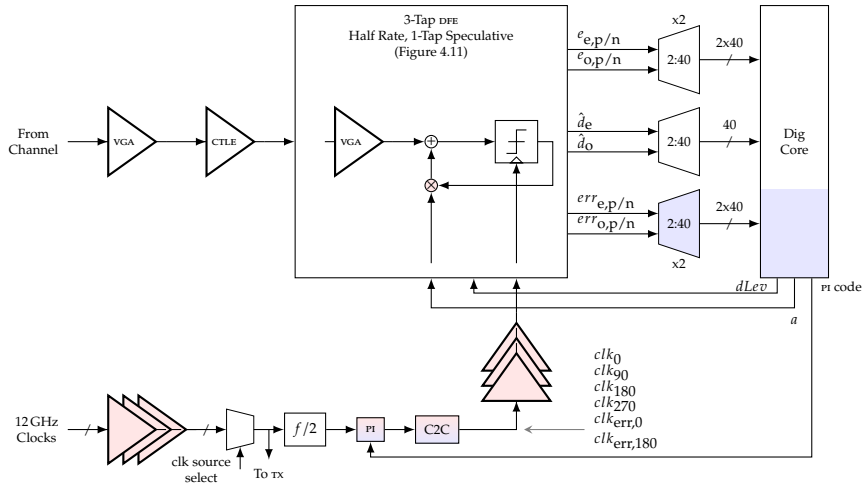


Figure 4.10: Simplified block diagram of the receiver of TC20 (refer to Figure 4.11 for more details on the DFE): The input signal from the channel is conditioned and equalized by the AFE (VGA and CTLE) and the DFE; data, edge and error samples are then deserialized and provided to the digital macro as 40-bit vectors; the digital circuit performs adaptation, eye monitoring, error checking and runs the CDR algorithm. Full-rate 12 GHz differential clocks from various sources are buffered into the chip, selected and divided by two for the half-rate receiver; then, its phase is adjusted according to the PI code determined by the CDR, converted into CMOS levels (C2C is the CML-to-CMOS converter) and buffered to the receiver. The blocks added for TC20 (the variable threshold comparators and the following deserializers, an additional PI and the following distribution circuitry for error sampling, and part of the synthesised logics) and those upgraded (the clock distribution, the PI, the CML-to-CMOS converter and the multipliers for the taps value a, actually implemented as current DACS) are shaded in blue and red, respectively; the input clock and the received signal up to the deserializers are differential.

opposite polarities of the clock (clk_0, clk_{90} and $clk_{\text{err},0}$ for the even half, clk_{180}, clk_{270} and $clk_{\text{err},180}$ for the odd half). Then, inside each half circuit, the DFE summing node and the following slicers are replicated to perform speculation: One of the replicas always adds $-a_1$ to the received signal, the other one always a_1 to consider both the possible symbols, whereas the other two taps (a_2 and a_3) are the same for both paths of the speculative architecture; the DFE-equalized signals y are then sliced and speculation is resolved through a multiplexer selected by the output data $\hat{d}$ of the other half circuit. Notice that a fourth "tap" is used for data slicers' offset correction through voltages V_{ov} (refer to Figure 4.18 for details on the circuit that applies the DFE correction). Edge samples e are also acquired through 90°-delayed clocks to provide information on transitions for the CDR. Error samples err are obtained through the variable-threshold comparator of Figure 4.12 driven by either $dLev$ or v_{err} for feeding the fully-adaptive algorithm or the eye monitor, respectively: In the former case, speculation is performed by replicating the procedure implemented for data samples, whereas in the latter case both p and n error samples are provided to the digital macro and speculation is resolved therein.

At the output of the DFE data, edges and errors are sampled and deserialized into

vectors of 40 bits and sent to the digital macro, where synthesised logics perform equalizer adaptation, eye monitoring, error checking and run the CDR algorithm presented in [109]; it then provides $dLev$ and the DFE taps a to the receiver, as well as the code words to drive the three PIs (see Figure 4.14a) for data, edge and error sampling. The full-rate 12 GHz differential clock is provided from the outside of the HSIO and is buffered into the chip to feed the transmitter, while it is divided by two to clock the half-rate receiver: The PIs adjust its phase according to the PI code determined by the CDR and generate all the six clock phases that are required (see Figure 4.11 for more details); such CML clocks are then converted to CMOS levels (see Figure 4.15a) and buffered before reaching the receiver.

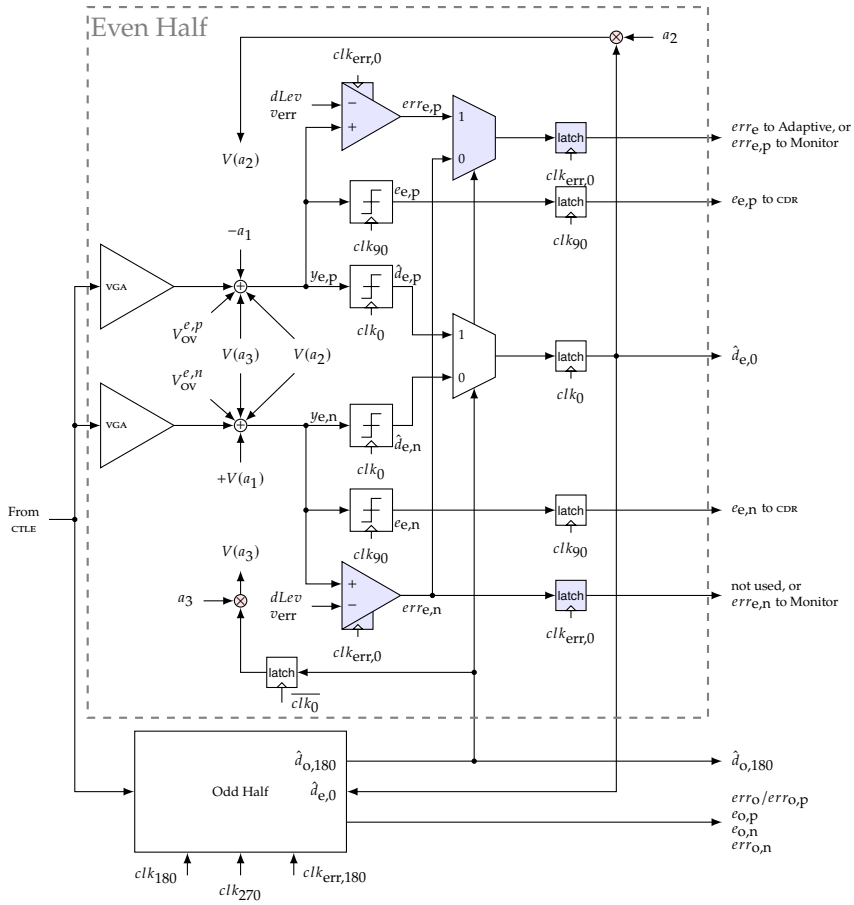


Figure 4.11: Schematic of the half-rate, 1-tap speculative DFE; only the even half circuit is shown explicitly, whereas the block representing the odd half is an exact replica of the one in the dashed gray box. The blocks added for TC20 (the variable threshold comparators, the following multiplexer and latches) and those upgraded (basically, the multipliers for the taps value a_n, actually implemented as current DACs) are shaded in blue and red, respectively; a signal $x_{a,b}$ belongs to the even/odd half receiver for $a = e/o$, respectively, and to the speculative DFE path that subtracts/adds the first tap for $b = p/n$, respectively.

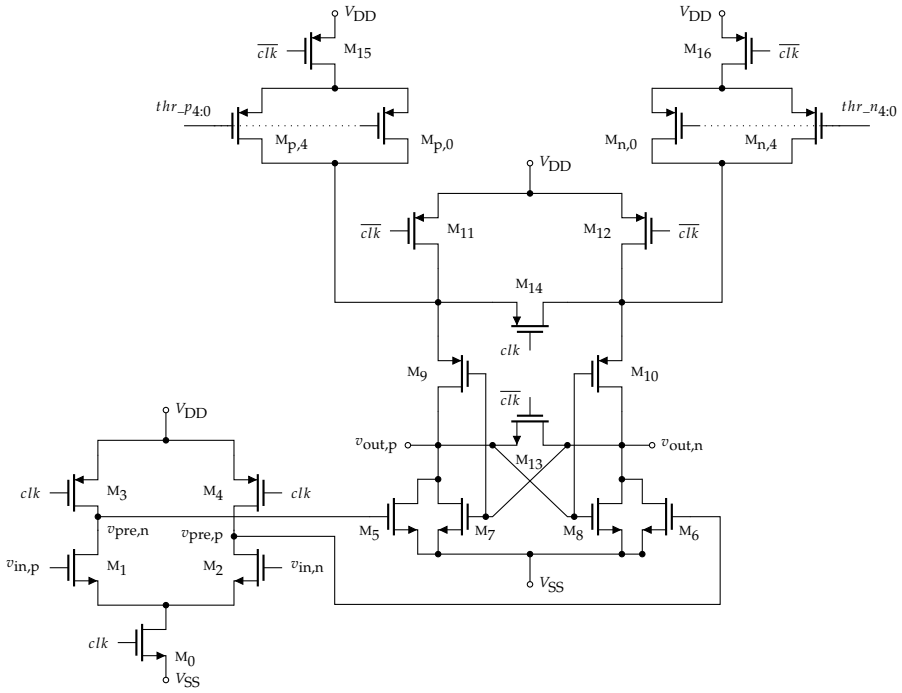


Figure 4.12: Schematic of the double-tail variable-threshold comparator employed as error sampler. (Adapted from [118].)

4.2.2.1 Variable-Threshold Comparators

Recalling the explanations on adaptive equalization of Chapter 3 and the scheme of Figure 3.1, the need for obtaining error samples from the received signal has to be met by implementing a variable-threshold comparator, whose inputs will be the incoming, equalized (after AFE and DFE) signal and the $dLev$ coming from the adaptive algorithm coded into the digital macro.

Such a comparator was designed by Dylan D'Ampolo from University of Udine and Infineon Technologies [118], and is reported herein to show its characteristics and main features. The chosen topology is reported in Figure 4.12 and is based on the double-tail, variable-threshold clocked comparator proposed in [119]. The first stage pre-amplifies the differential input signal $v_{\text{in,p}} - v_{\text{in,n}}$ when the clock is low, whereas the second stage regenerates the voltage $v_{\text{pre,n}} - v_{\text{pre,p}}$ through the two cross-coupled pairs M_7, M_8 and M_9, M_{10} when the clock is high. The output differential signal $v_{\text{out,p}} - v_{\text{out,n}}$ eventually saturates to the voltage supplies and is subsequently driven into an SR latch (not shown in the schematic) to retain its state; this last step is required because such a circuit is reset every clock period to avoid memory effects, so that otherwise its output voltage would effectively be valid for less than half a period (the time when it is active minus the time that the output takes to reach a defined logic value).

The threshold of the comparator is programmed through the four-bit control words thr_p and thr_n by activating a number of the five PMOSS in any of the two branches of the second stage featuring threshold transistors $M_{p,4–0}$ and $M_{n,4–0}$,

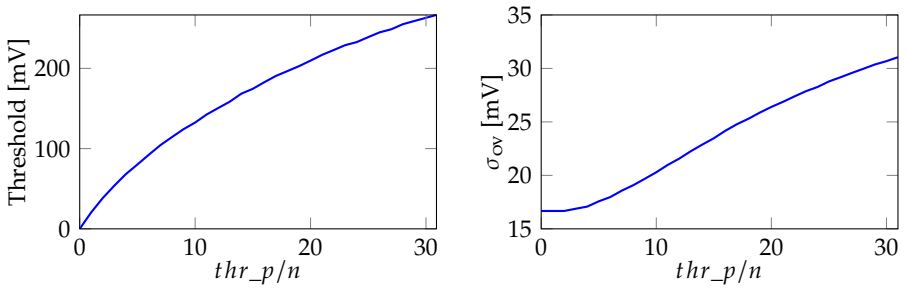


Figure 4.13: Post-layout simulations of the variable-threshold comparator: (a) shows the thr_p/n-threshold voltage characteristics, (b) reports Monte Carlo estimation of the standard deviation σ_{ov} of the input offset voltage as a function of the digital threshold code; all voltages are differential. (Adapted from [118].)

which are connected in parallel and whose sizes are scaled in a binary fashion. When none of them is enabled, the second stage of the comparator is symmetric and it simply regenerates the voltage difference between its inputs, determining when it is larger or smaller than 0 V; when any of the threshold transistors is enabled, the two branches of the regeneration stage are unbalanced because more current flows through one of them, effectively working against discharging the corresponding output node. This is equivalent to adding a voltage to one of the second stage's inputs, so that the comparator's output toggles at a value of differential input different than 0 V, thus changing the threshold.

The comparator was simulated after layout and its main features at nominal operating conditions (40 °C, V_{DD} = 0.95 V and nominal technology corner) are reported in Figure 4.13. The differential threshold voltage characteristic is shown in Figure 4.13a, displaying a non-linear trend reaching a maximum value of ≈ 265 mV, whereas Figure 4.13b presents the standard deviation σ_{ov} of the differential input offset voltage obtained by running Monte Carlo simulations on each threshold code.

Four such variable-threshold comparators have to be instantiated in parallel to the data slicers already present in the HSSI, due to the half-rate architecture of the receiver and to the one-tap speculative topology of the DFE (see Figure 4.11). Moreover, in order for the adaptive equalization algorithm to optimise the sampling phase, an additional phase interpolator has to be instantiated, driven by the code produced by the CDR with the offset provided by the SS-LMS loop. This requirement provided the opportunity to modify the PI and other critical circuits in the clock distribution, as detailed next.

4.2.2.2 Clocking

The most significant upgrades in the clock distribution network (shown in Figure 4.10) were all devoted to improve robustness to PSIJ and were performed in collaboration with Dylan D'Ampolo from University of Udine and Infineon Technologies. As already mentioned, the PI was modified by resizing the current mirrors that provide I_1 in Figure 4.14, so to increase the PI's gain and thus steepen

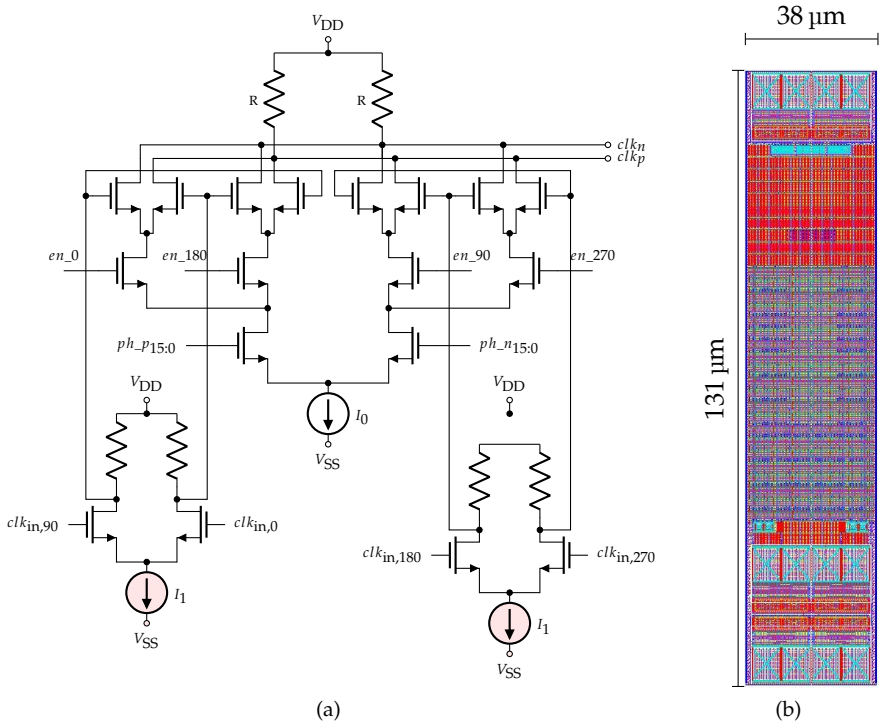


Figure 4.14: (a) Simplified schematic and (b) layout (not on scale) of the phase interpolator. The upgraded elements are shaded in the schematic.

its output CML edges, in turn reducing its sensitivity to PSIJ, although to reduce layout and floorplanning efforts the choice was made to modify the circuit only to the extent that the modification did not increase area consumption.

The CML to CMOS converters (Figure 4.15) that follow the PIS and effectively provide the clock to the buffers leading to the slicers were also modified by increasing the decoupling capacitors of the input AC-coupled inverters and by resizing all their inverters, so to enhance their ability to restore the clock's edges even in the presence of supply noise.

Additionally, the whole clock distribution network from the input of the SerDes to the samplers was reconsidered to minimise the accumulation of jitter induced by supply noise and to comply with the clock sources in TC20. With respect to TC18, the new chip can use two different clock sources placed outside the SerDes itself: A digital PLL situated on the same chip and an off-chip clock generator for enhanced versatility and easier bring up in the laboratory; before reaching the HSIO, the CMOS differential clock coming from the PLL is simply buffered, whereas the external source requires also a clock receiver, which is implemented as a cascade of two AC-coupled inverters, having the same topology of the first stage of the CML-to-CMOS converter of Figure 4.15a. Then, as shown in Figure 4.10, inside the transceiver the clocks are buffered and fed to a multiplexer so that the user can select the desired clock source to feed both transmitter and receiver; the clock going to the latter block is frequency-divided by 2 to comply with the

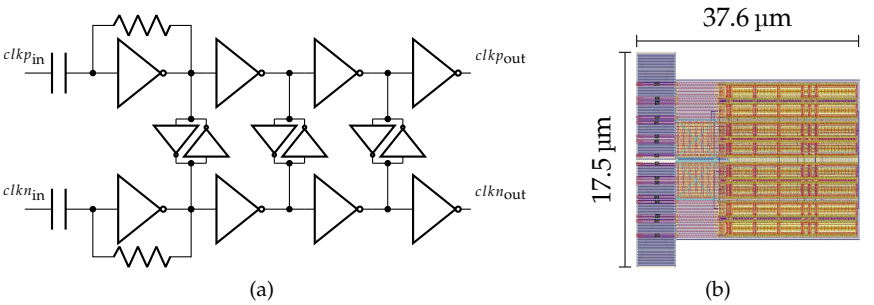


Figure 4.15: (a) Schematic and (b) layout (not on scale) of the CML-to-CMOS converter.

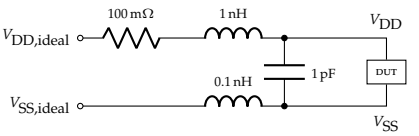


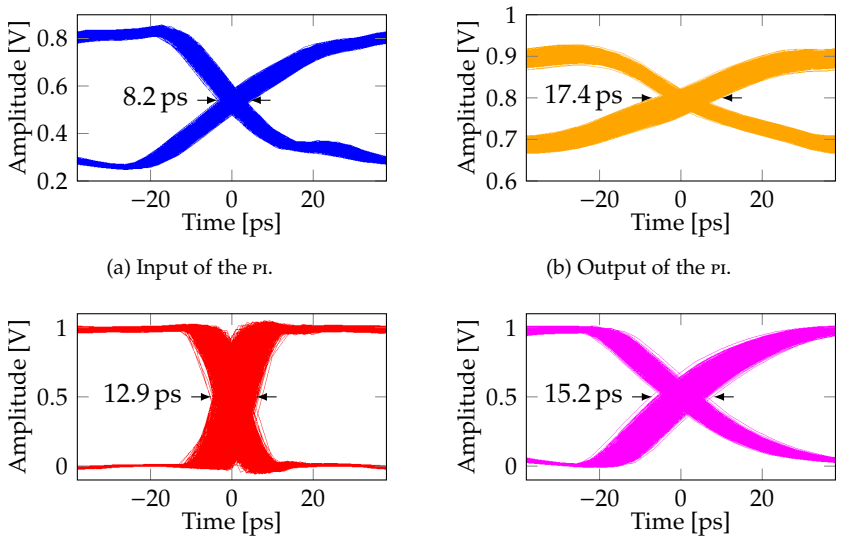
Figure 4.16: power delivery network used to simulate jitter in the clock distribution network.

half-rate architecture of the receiver and results in a CML clock for the PI, which drives a CML-to-CMOS converter to restore rail-to-rail clock signals; eventually, the resulting clocks are buffered and distributed to the data path.

The complete post-layout clock distribution network was then simulated extensively with the whole transceiver and the *power delivery network (PDN)* shown in Figure 4.16 ($R = 100\,\text{m}\Omega$ and $L = 1\,\text{nH}$ on V_{DD}, $L = 0.1\,\text{nH}$ on V_{SS} and $C = 1\,\text{pF}$ as external decoupling capacitor) to verify its behaviour regarding PSIJ and jitter performance. Figure 4.17 reports eye diagrams of the single-ended clocks[3] in various positions along the clock chain with the corresponding peak-to-peak jitter extracted from simulations, in presence of approximately 60 mV and 30 mV peak-to-peak voltage noise[4] on V_{DD} and V_{SS}, respectively: Starting from an ideal albeit sinusoidal source, the CML clock at the input of the PI (Figure 4.17a, right after frequency division) shows a jitter of 8.2 ps, which increases to 17.4 ps in Figure 4.17b after phase interpolation; conversion to CMOS reduces the jitter to 12.9 ps (Figure 4.17c), whereas the slicers are fed with a clock that features a 15.2 ps peak-to-peak jitter.

[3]Despite being generated and treated as differential signals all the way through the clock distribution network, they are reported here as single-ended signals because this is how they are actually used to clock most of the digital circuits in the mixed-signal portion of the wireline transceiver, chiefly data and edge slicers.

[4]Supply noise is due to the circuit's switching activity only, no other noise sources were added to the simulation.



(a) Input of the PI.

(b) Output of the PI.

(c) Output of the C2C converter.

(d) Clock at the samplers

Figure 4.17: Post-layout simulation eye diagrams of the single-ended, 13 GHz clock along the distribution network: at the (a) input and (b) output of the PI, (c) after the CML-to-CMOS converter and (d) at the input of the slicers.

4.2.2.3 DFE

The circuit that implements DFE was also modified to comprise the additional slicing and multiplexing required by equalizer adaptation and eye monitoring, as shown in Figure 4.11.

The implementation of the fully-adaptive equalization algorithm provided the opportunity to also enhance the granularity of the DFE correction, since an automatic procedure to set such weights can take advantage of this to yield a more accurate equalization, whereas it might result impractical when selecting the tap values manually. The current digital-to-analog converters (IDAC) shown in Figures 4.18a and 4.18b that implement the tap weights were extended from 4 to 5 bits while maintaining the same full scale range as the previous version, thus achieving a nominal least significant bit (LSB) amplitude of 4.9 µA (vs. the nominal 9.8 µA, effectively ≈ 14 µA, of the old version); when applied with a circuit similar to that shown in Figure 4.18c (featuring $R = 500\,\Omega$), this translates to a nominal LSB of 2.45 mV vs. 4.9 mV of TC18.

4.2.2.4 Digital Eye Monitor

A novel feature of TC20 is the digital eye monitor for assessing the quality of the incoming signal at the slicers' input. This allows qualitative evaluation of the eye diagram in a position that would not even be reachable for measurements, thus providing valuable information on degradation due to ISI and jitter, or on the behaviour of the analog front end; moreover, ~~by~~ monitoring of the actual BER eye~~, it~~ can also be used to drive adaptation, mainly through heuristic techniques [23, 99, 120].

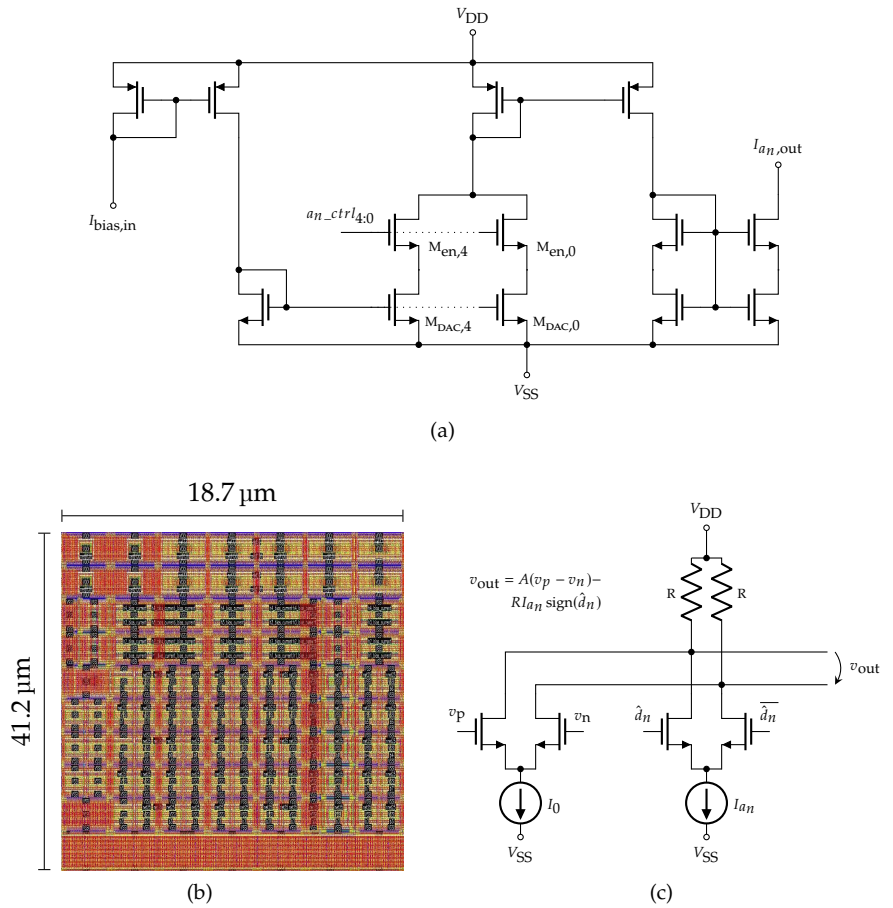


Figure 4.18: (a) Schematic and (b) layout (not on scale) of one of the four IDACS that implement the DFE tap weights; (c) shows a simplified schematic representing the application of the DFE tap correction on the received analog voltage (after the VGAS shown in Figure 4.11, represented as a simple differential pair), where the current source I_{a_n} represents the schematic of subfigure (a).

The implemented eye monitoring procedure is based on [120], where the eye diagram is scanned by sweeping all its width and height through an error sampler driven by a variable threshold v_{err} (as the one depicted in Figure 3.1, albeit with $dLev$ replaced by such a threshold). For each sampling phase φ_{err}, the variable threshold is swept in its entire range and for each point (φ_{err}, v_{err}) the number of detected errors equal to '1' (i.e. when the equalized signal is larger than the current threshold, always computed for the same value, either 1 or 0, of the corresponding data bit), N_{e}, is accumulated. Supposing that the threshold's voltage range exceeds that of the eye amplitude, when v_{err} is either $v_{\mathrm{e,min}}$ or $v_{\mathrm{e,MAX}}$ (the actual values depend on the implemented comparator) all errors will be 1 and 0, respectively; when $v_{\mathrm{err}} = 0\,\mathrm{V}$, errors and data match so that N_{e} equals the number of received data bits of the observed data value (either 1 or 0), $N_{\mathrm{d=1/0}}$; when the threshold equals the voltage corresponding to the observed data value (basically, $dLev$ when observing the 1) in the eye diagram, half the errors are statistically expected to be

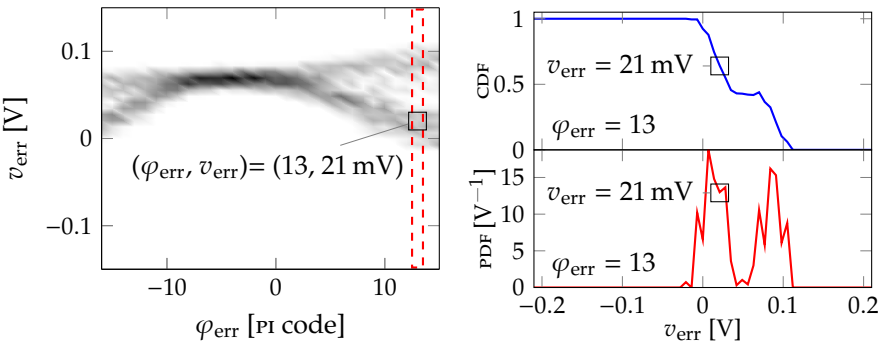


Figure 4.19: Working principle of the implemented eye monitor, based on [120]. The top half of an eye diagram is shown in plot (a), highlighting the slice pertaining to a certain sampling phase φ_{err}, whereas plot b reports the corresponding CDF ($\approx N_{\text{e}}/N_{\text{d=1/0}}$) and PDF ($[\text{CDF}(v_{\text{err}}) - \text{CDF}(v_{\text{err}} - \Delta v)]/\Delta v$) when the observed data value equals 1.

0, the other half 1, so that $N_{\text{e}}/N_{\text{d=1/0}} \approx 1/2$. By generalising such a principle to the whole range of v_{err}, it can be seen from Figure 4.19 that such a procedure generates a *cumulative distribution function (CDF)* for each sampling phase φ_{err}, whose first derivative yields a *probability density function (PDF)* that effectively describes the density of vertical slices of the eye diagram as would be seen on an oscilloscope. The full eye diagram is then obtained by iterating this procedure over the whole UI, one per each of the observed data value (1 and 0), so to obtain the two halves (top and bottom) of the eye.

Figure 4.19 reports an example of the top half of an eye diagram (hence acquired when data bits are 1) and the CDF and PDF corresponding to a specific sampling phase highlighted in plot (a). Since such an eye monitor relies on a statistical approach, and accumulation for this simple example was performed on a reduced number of bits per eye point (φ_{err}, v_{err}) to speed up the simulation, it results in a non-monotonic CDF and in an unsmooth PDF; obviously, the quality of this procedure improves by observing more data bits per eye point, as is done in the actual implementation.

The scheme of the proposed implementation is reported in Figure 4.20 and exploits the same variable-threshold comparators implemented for the error sampling in the adaptive equalization algorithm, so to avoid instantiating more components that would add load to the clocks. Additional considerations on minimising analog hardware overhead and additional clock loading linked to the eye monitoring procedure led to the decision to deserialize the error data from both the unrolled DFE paths and to resolve speculation in the digital domain, despite at the cost of one additional deserializer (while the one instantiated for the fully-adaptive algorithm is reused). Therefore, the digital macro receives two 40 b error vectors (one for each polarity of the loop-unrolled DFE) which are processed at low frequency ($T_{\text{B}}/40$) to resolve DFE speculation by checking the received data vector, in fact reproducing the procedure employed in the analog domain for the equalized data (see Figure 4.1).

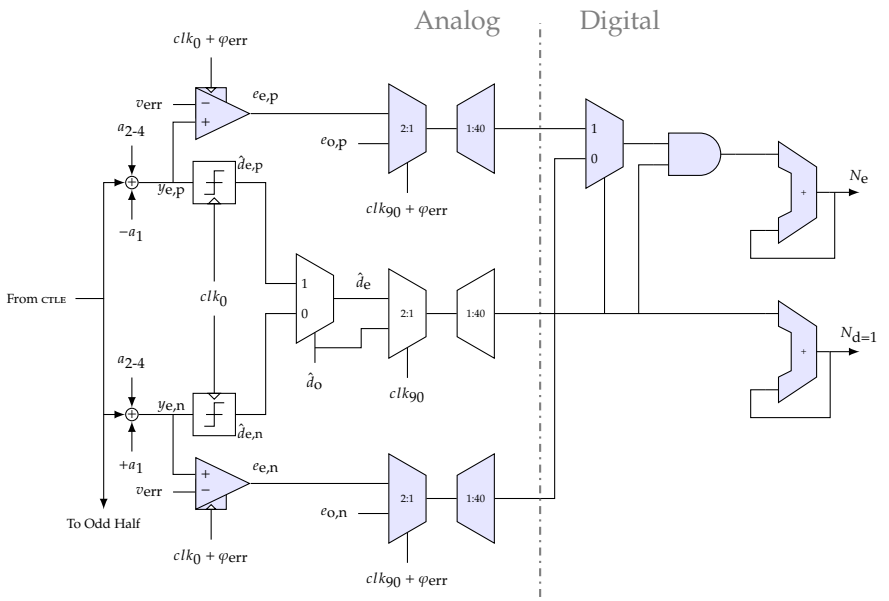


Figure 4.20: Simplified block diagram of the digital eye monitor implemented in TC20 to allow the visualisation of the effect of the full DFE, despite the first tap being speculative. Only the even half of the analog circuit is shown for simplicity and only the case when data bits are equal to '1' is taken into account in the digital circuit (for the other case, it is sufficient to invert the deserialized data bits entering the digital macro); a signal $x_{a,b}$ belongs to the even/odd half receiver for $a = e/o$, respectively, and to the speculative DFE path that subtracts/adds the first tap for $b = p/n$, respectively; N_e is the number of errors (i.e. cases with $y_{a,b} - v_{err} > 0$), $N_{d=1}$ is the number of received data bits equal to '1'; shaded blocks are specific to the digital eye monitor, whereas the others are involved in regular data reception (as in Figure 4.1).

As can be seen in Figure 4.20, the proposed eye monitor differs significantly from [120] in the way the acquired information is processed: Instead of low-pass filtering the output of the error comparator and converting it into a digital word, in the proposed implementation the total numbers N_e and $N_{d=1/0}$ for each eye point are computed in the digital macro and provided externally to be processed on a regular computer. This increases area occupation and power consumption of the digital circuits, but reduces the dependence of the eye monitor on analog circuits, basically limited to the variable-threshold comparator.

The control logic is coded in VHDL to be synthesised and takes care of automatically sweeping both v_{err} and φ_{err}, observing each eye point for a sufficient amount of time to reach the user-requested BER level, while there is also an option to perform the scanning procedure manually for debugging purposes; the synthesised logic is also responsible for the handshake required to communicate the acquired result to the computer controlling the test chip in the laboratory sessions.

A practical simulated example is reported in Figure 4.21 and shows eye diagrams computed on the behavioural model of the HSIO (refer to Section 4.3 for further information on this) after a channel losing 14.3 dB at 6 GHz (see Figure 4.24a),

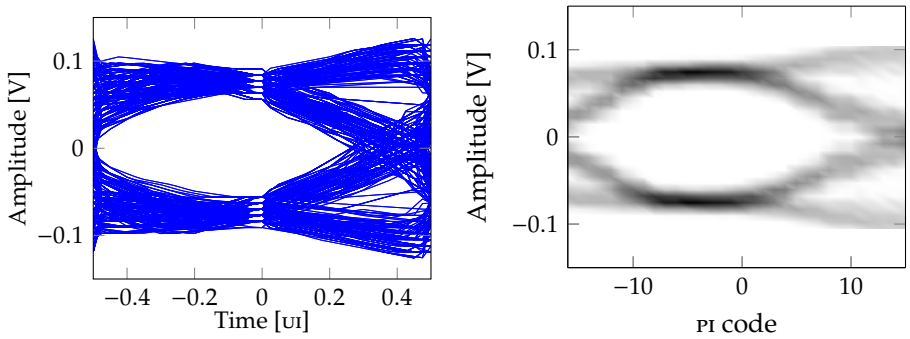


Figure 4.21: Simulation of a channel with 14.3 dB attenuation at 6 GHz (see chA of Figure 4.24a); eye diagrams at 12 Gb/s, on transmitted pulses with 10 % effective rise/fall time and ±0.28 V differential voltage swing (equivalent to 8 driver slices connected at the transmitter); FFE was fixed at $w_{\text{-}1} = -4/16$ and $w_0 = 12/16$, while the receiver's VGAs and CTLE were set as buffers. Both plots are obtained on the behavioural SerDes (a) with the virtual eye monitor of Figure 3.13, whereas (b) is computed by the VHDL code implementing the digital eye monitor of Figure 4.20.

FFE with ≈ -6 dB de-emphasis and DFE featuring $a_1 = 21$ mV, $a_2 = 7$ mV and $a_3 = 7$ mV. Figure 4.21a is computed from the virtual eye monitor of Figure 3.13, whereas the eye of plot (b) is obtained by simulating the actual digital eye monitor code. The qualitative correspondence is rather accurate, whereas quantitatively the difference in eye amplitude can be explained by the voltage values assumed for the threshold of the error comparators: While the virtual eye monitor mimics the behaviour of analog circuits, hence visualising the actual voltages at the samplers' input, the digital eye monitor requires a map between the threshold codes and the actual voltages that they represent to provide a vertical scale to the resulting eye diagram; as was shown in Figure 4.13a, such a characteristic is in fact not linear so that, if a linear map were assumed, the eye diagram amplitudes would differ from the actual values, as is the case in Figure 4.21b.

4.3 TC20: RTL Implementation of Fully-Adaptive Algorithms

The final step in the development flow consisted in "porting" the behavioural code into VHDL, since such a version written in a hardware-description language can be synthesised and then fabricated on a chip. Therefore, not only did this step allow to work with the final version of the fully-adaptive algorithm, it also enabled simulating it along with the combined analog and digital parts of the HSSI, thus verifying its overall correct functioning.

Implementing the fully-adaptive algorithm in a RTL description required careful considerations regarding the internal representation of the various signals and registers, in the sense that choices had to be made on matters such as the width of the buses used to internally compute the updates or on the range available for parame-

ters such as the step sizes μ_c of Equation 3.18. Therefore, once a first version of the code was completed, extensive tests were carried out in order to determine useful ranges for the aforementioned parameters to be made available to the final user.

A simplified block diagram of the mixed-signal hardware coupled to the RTL implementation is shown in Figure 4.22. Since the digital macro cannot work on the full-rate data, but acquires deserialized 40 b vectors from the analog SerDes instead, the minimum update time of the digital SS-LMS algorithm is $N_{\text{des}}T_{\text{B}}$, where $N_{\text{des}} = 40$ is the (fixed) deserialization ratio of the receiver. Moreover, in order for the adaptation loop to accumulate enough samples so to observe the effect of a sufficient number of bit sequences, a variable (user-programmable) number N_{mean} of 40 b vectors is gathered before updating the outputs, so that the effective update time is $N_{\text{mean}}N_{\text{des}}T_{\text{B}}$.

Time-averaging and multiplication by μ_c in Figure 4.22 is performed through the final shifter before the DAC, which is not necessarily an actual circuit because the adapted parameter c may be provided as a digital word and applied/converted directly in the corresponding circuit (e.g. see Figures 4.12 and 4.18a for how $dLev$ and a_n are applied). The actual value of N incorporates μ_c as well as the number

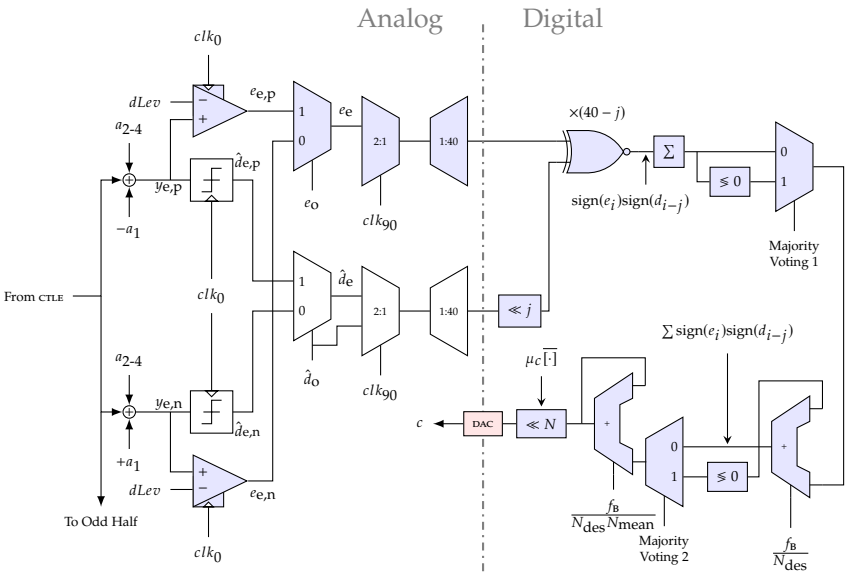


Figure 4.22: Simplified block diagram of the *register-transfer level (RTL)* implementation of the fully-adaptive algorithm. Only the even half of the analog circuit is shown for simplicity; $\ll$ represents shifters, $\lessgtr 0$ performs majority voting; a signal $x_{\text{a,b}}$ belongs to the even/odd half receiver for $a = e/o$, respectively, and to the speculative DFE path that subtracts/adds the first tap for $b = p/n$, respectively; blue-shaded blocks are specific to the adaptation circuitry, whereas the others are involved in regular data reception (as in Figure 4.1); the DAC (shaded in red) is fictitious, i.e. it is not necessarily an actual separate circuit, because adapted parameters ($dLev$, DFE taps and φ_{shift}) can be provided as digital words to the analog macro and therein applied/converted directly inside the respective circuits (e.g. see Figures 4.12 and 4.18a for how $dLev$ and a_n are applied); the registers after the accumulators are not drawn for simplicity.

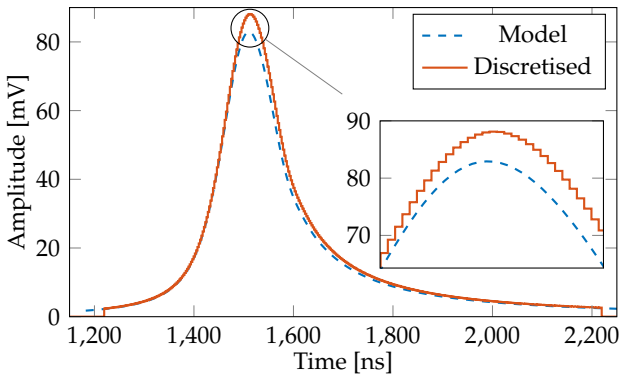


Figure 4.23: Pulse response obtained with the model of Section 2.2.3 for a 12 Gb/s transmitted pulse with 0.28 V single-ended swing and the resulting discretised waveform used with the behavioural models in the digital simulator; the inset displays a magnified version of the pulses' peaks, showing a small discrepancy of a few mV in amplitude.

of samples accumulated before updating the parameter c, i.e. $N_{\text{mean}}N_{\text{des}}$ unless one or more of the majority voting blocks are enabled. Such voters are equivalent to performing a sign$(\cdot)$ operation, thus allowing the following accumulators to operate on single-bit increments/decrements only (as is often the case in CDRs, see Appendix A), and were inserted in the design to study such alternatives on chip. Needless to say, in order for the digital hardware to perform such operations easily, the aforementioned control parameters are programmable as powers of 2 only.

Verification of the RTL code was carried out in a digital simulation environment, along with the whole digital part of the HSSI and a behavioural description of the analog circuit which, as such, is in fact an idealised version of the actual circuitry. The latter had to be modified in order to correctly simulate the adaptive algorithm and its effect on the interface: Among the main modifications, the transmitted signals were represented as real numbers corresponding to voltages, hence allowing proper simulation of the equalization process, otherwise unfeasible in a purely digital environment; and a behavioural model of the channel was implemented in order to apply the desired ISI through convolution with the pulse response obtained from the model described in Section 2.2.3. This latter aspect required to discretise the temporal resolution of the pulse response in order for the digital simulator to perform the convolution with the transmitted data sequence; aiming at reducing the overhead in simulation runtime, the pulse response was discretised so that an amount of samples equal to the number of PI steps is available on every UI, thus ensuring that each different sampling position determined by the CDR results in a different "analog" voltage at the slicers' input, whereas a finer resolution would not increase the simulation accuracy, because practically not reachable. This in turn implies that the pulse response's amplitude is discretised (as can be seen in Figure 4.23), otherwise more involved simulation approaches would be required to extend the accuracy below such a limit [121].

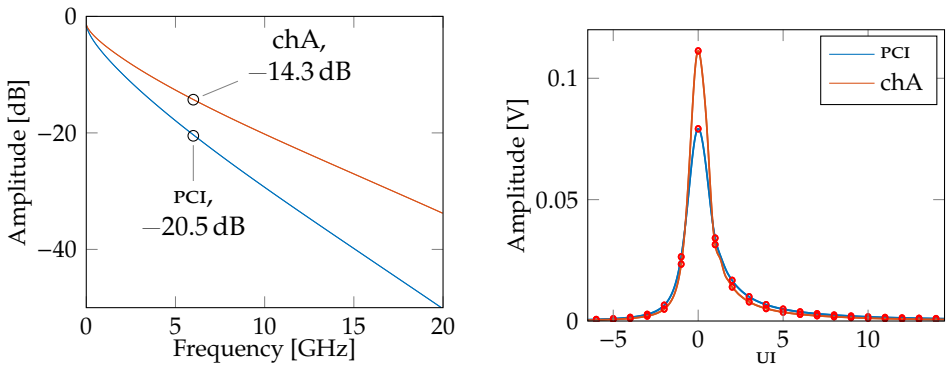


Figure 4.24: Simulated (a) frequency response and (b) corresponding pulse response to a trapezoidal pulse for the PCI 4.0 channel [122] attenuating 20.5 dB at Nyquist frequency (blue line) and for chA attenuating 14.3 dB at the same frequency (red line). With both channels, the trapezoidal pulse was sent at 12 Gb/s bit rate, with 10 % rise time and ±0.28 V differential voltage swing.

4.3.1 Comparison of the Three Versions

In order to validate the work on the SS-LMS fully-adaptive algorithm, a comparison was performed among the various implementations: The probabilistic model written in MATLAB of Section 3.3.1, the behavioural version working on the transistor-level SerDes of Section 3.3.2 and the RTL code simulated on the behavioural description of the HSIO. As a test channel, PCI 4.0 specifications were considered [122] to derive a channel model for transmission at 12 Gb/s, which is the same speed used in the simulations of [76] and of Section 3.3.2. However, the standard only specifies maximal insertion losses of 22 dB and 23.5 dB at Nyquist frequencies of 4 GHz and 8 GHz, respectively, entailing that the fastest data rate is not necessarily capable of transmitting on the worst-case channels used for the slower gears (unless a channel is found such that its loss decreases less than 4 dB/octave in frequency, which should not be the case for realistic skin effect- or dielectric losses-dominated channels); therefore, the channel specified for the 16 Gb/s data rate was chosen so to perform the test in a realistic application scenario (even though with a lower loss than that of the actual 16 Gb/s case), with the resulting insertion loss reaching 20.5 dB at the Nyquist frequency of interest (6 GHz).

Another test channel, named *chA*, with a lower attenuation of 14.3 dB at 6 GHz was considered in order to test the SS-LMS algorithm in a situation in which channel losses are not the dominant phenomena, so that the performance of the fully-adaptive procedure can be evaluated more effectively. The resulting unequalized frequency and pulse responses obtained with the method of Section 3.3.1 for both the aforementioned channels are shown in Figure 4.24: The numerous relevant cursors (with respect to the main cursor) in the PCI channel imply a closed eye diagram (see Figure 4.25a), which in turn requires equalization to be applied; the other test channel, despite the large number of non-zero cursors, has a much larger main cursor, so that the effect of ISI is less relevant in this case and the eye diagram at the receiver displays a small aperture even without FFE, not shown in Figure 4.25b.

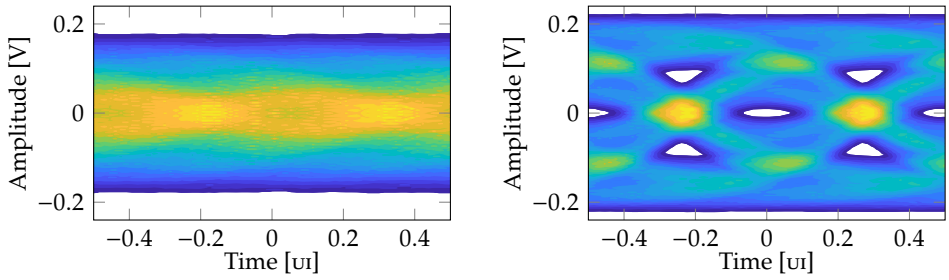


Figure 4.25: Simulated unequalized eye diagrams of (a) the PCI 4.0 channel attenuating 20.5 dB at Nyquist frequency (b) chA attenuating 14.3 dB at the same frequency. With both channels, the trapezoidal pulse was sent at 12 Gb/s bit rate, with 10 % rise time and ±0.28 V differential voltage swing.

The three different implementations of the algorithm were run with the same parameters: Averaging over $N_{\text{mean}} = 8$ vectors of $N_{\text{des}} = 40$ bits, $\mu_{dLev} = 7\,\text{mV}$, $\mu_{\text{DFE}} = 3.5\,\text{mV}$ and $\mu_{\varphi} = 2.8125°$. The HSSI was simulated at 12 Gb/s, transmitting pulses with 10 % rise/fall time and ±0.28 V differential voltage swing (8 driver slices connected at the transmitter)[5]; both channels were tested with a fixed FFE de-emphasis having $w_{\text{-1}} = -4/16$ and $w_0 = 12/16$, while the receiver's VGAS and CTLE were set as buffers.

The results for the PCI channel are shown in Figure 4.26, where the numerical algorithm provides the pulse responses without equalization, with only FFE and with fully-adaptive equalization in Figure 4.26a, showing the effect of the optimization of both sampling phase and DFE; Figures 4.26b to 4.26d show the resulting eye diagrams computed with the numerical, behavioural and RTL versions of the algorithm, respectively, the latter two exploiting the virtual eye monitor of Figure 3.13; Figure 4.27 displays analogous results for chA. In order to avoid confusion in the following, recall that the behavioural version of the algorithm is simulated with the transistor-level circuit, while the RTL version runs on the behavioural description of the analog hardware.

Figure 4.26a shows that the fully-adaptive procedure is very effective in eliminating ISI from the pulse response: The first three post-cursors are practically null thanks to the DFE, while the first pre-cursor is driven to zero by adapting the sampling phase. The qualitative correspondence among the eye diagrams is rather good, although quantitatively some relevant differences appear, chiefly in the eye aperture resulting from the RTL version (which is smaller than the other two) and in the overall amplitude of the probabilistic version (which is slightly lower than the other two). Some considerations have to be made in this respect, especially the fact that the behavioural version is simulated with the full post-layout transistor-level implementation of the HSSI, entailing that all non-idealities of the analog circuitry are taken into account, e.g. random jitter, full frequency response of the various blocks, etc. The RTL algorithm, instead, was simulated with a behavioural version

[5]The behavioural HSIO cannot model driver's rise/fall times and the effect of varying connected slices; however, the rise/fall time of the transmitted pulse is embedded in the channel model that implements convolution with the pulse response, whereas the number of driver slices is modelled simply as output amplitude, so that such parameters have to be construed as *effective* parameters.

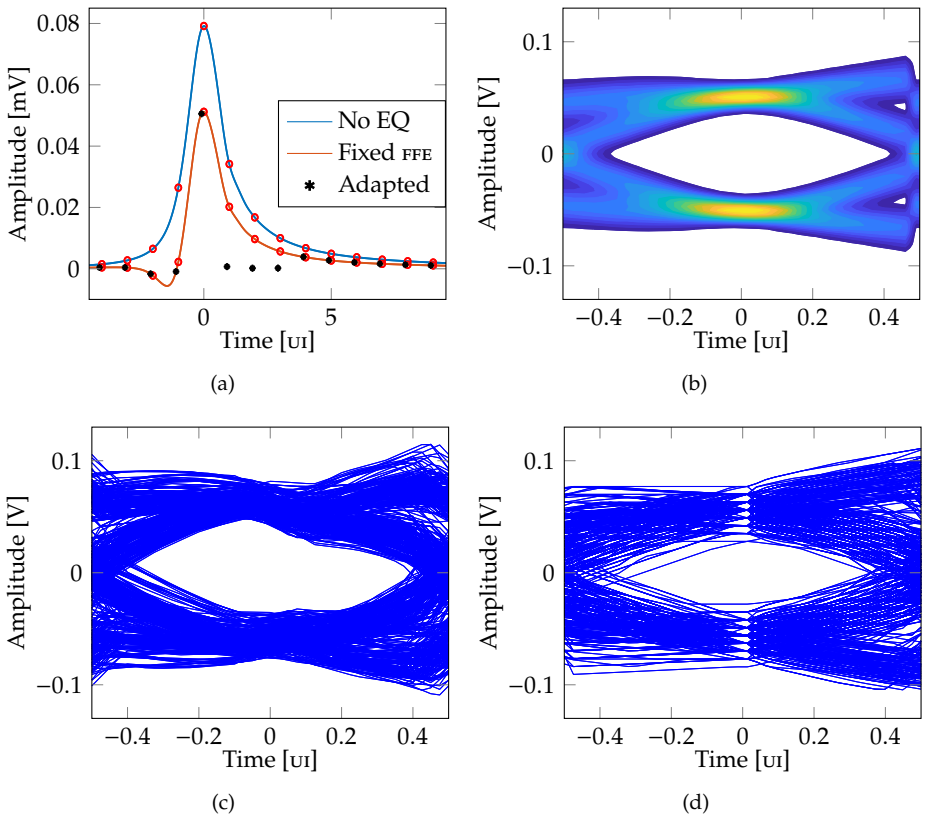


Figure 4.26: Simulation of a PCI 4.0 channel with 20.5 dB attenuation at 6 GHz (see Figure 4.24a). (a) Shows the pulse response obtained in Matlab with the method of Section 3.3.1 prior to equalization (solid blue line), after the fixed 2-tap FFE (solid orange line) and after application of the fully-adaptive loop and subsequent sampling (black asterisks); (b) shows the resulting eye diagram obtained with the same numerical method; (c) depicts the eye diagram obtained with post-layout transistor-level simulation and behavioural adaptive algorithm as in Section 3.3.2; (d) shows the eye diagram obtained with digital simulation considering a behavioural model of the HSSI, as in Section 4.3. The latter two plots were obtained exploiting the virtual eye monitor of Figure 3.13.

of the analog circuitry, hence neglecting many non-ideal effects and rendering the simulation more "deterministic", while the channel pulse response had to be relevantly quantised in order to work in the digital environment; the combination of the two previous considerations is likely to be responsible of the voltage levels splitting which is well visible at the sampling instant in Figures 4.26d and 4.27d.

Similar considerations can be drawn from the results of Figure 4.27 regarding the chA, losing 14.3 dB at 6 GHz. A peculiar difference with respect to the previous case can be noted in Figure 4.27a, where the sampling phase adaptation has still driven the first pre-cursor to zero, this time moving it towards the starting position of the main cursor, not away from it: This is due to the fact that the fixed FFE applied at the transmitter overly de-emphasised the trapezoidal pulses trans-

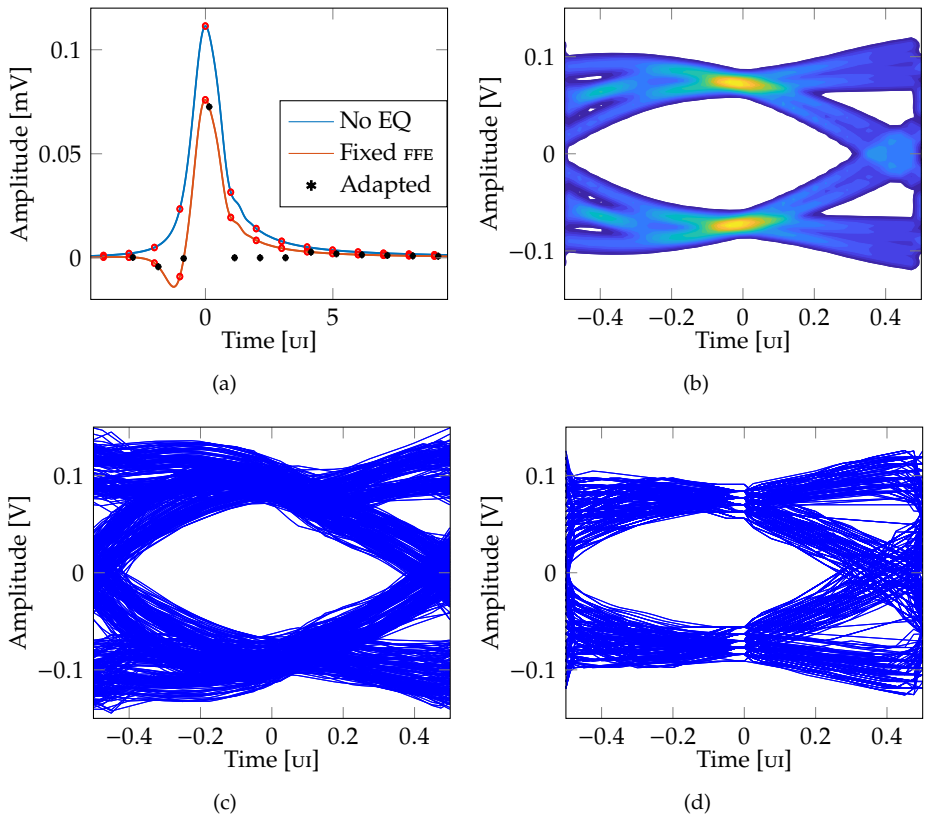


Figure 4.27: Simulation of chA featuring 14.3 dB attenuation at 6 GHz (See Figure 4.24a). (a) Shows the pulse response obtained in Matlab with the method of Section 3.3.1 prior to equalization (solid blue line), after the fixed 2-tap FFE (solid orange line) and after application of the fully-adaptive loop and subsequent sampling (black asterisks); (b) shows the resulting eye diagram obtained with the same numerical method. (c) depicts the eye diagram obtained with post-layout transistor-level simulation and behavioural adaptive algorithm as in Section 3.3.2; (d) shows the eye diagram obtained with digital simulation considering a behavioural model of the HSSI, as in Section 4.3. The latter two plots were obtained exploiting the virtual eye monitor of Figure 3.13.

mitted. With such a lower attenuation, the similarity between the eyes resulting from the numerical algorithm and the RTL version is evident, both qualitatively and quantitatively; the eye obtained with the behavioural algorithm looks quite similar to the other two, but shows a rather larger amplitude.

The convergence of the various SS-LMS parameters for the two channels considered is reported in Table 4.5, along with some significant figures of merit which are useful to evaluate the adaptive procedure. The SS-LMS parameters are reported as their average value after convergence. The sampling phase shift φ_{shift} is computed in different ways depending on the version of the algorithm: The numerical one starts evaluating the optimal sampling phase from the middle point between the two samples 1 UI away from each other that have equal amplitude (i.e. $h_{\text{-0.5}} = h_{0.5}$,

Parameter	PCI 4.0 channel Algorithm version			Channel chA Algorithm version		
	Num	Beh	RTL	Num	Beh	RTL
$dLev^a$ [mV]	50.9	60.8	51.9	72.3	89.5	69.7
DFE: a_1^a [mV]	21.5	38.0	26.3	17.3	23.7	17.3
DFE: a_2^a [mV]	10.1	15.4	12.2	7.6	11.2	7.1
DFE: a_3^a [mV]	5.7	9.0	3.5	4.2	6.6	3.4
φ_{shift}^a [UI]	-0.09^b	-0.07^c	-0.09^d	0.07^b	0.06^c	0.07^d
Convergence time [ns]	3470	3300	500	4270	1750	450
Eye height [mV]	74	87.5	56	114.2	142	117.7
Eye width [UI]	0.80	0.79	0.66	0.80	0.78	0.75

[a] Average value after convergence
[b] Computed with respect to the point for which $h_{-0.5} = h_{0.5}$ holds
[c] Computed with respect to the initial position
[d] Computed with respect to the CDR locking point

Table 4.5: Simulations at 12 Gb/s over the channels of Figure 4.24. Comparison of fully-adaptive algorithm parameters after convergence and figures of merit of the resulting eye diagrams for the various algorithm versions presented in this Section are reported. Recall that the behavioural version of the algorithm is simulated with the transistor-level circuit description, while the RTL algorithm version runs with the behavioural description of the analog hardware.

basically, emulating an Alexander phase detector [75]), which is easily computed in Matlab; the behavioural version does not show an exact correspondence between the algorithm's internal starting value and the initial, unequalized eye diagram, therefore the final phase shift may only be approximately estimated by comparing the edge of the sampling clock to the centre of the unequalized eye diagram, determined as the midpoint between transitions; the RTL version, instead, adds the computed optimal phase to the value provided by the CDR, hence allowing to determine such a shift with respect to the centre of the eye as determined by the CDR itself.

Table 4.5 also reports convergence times of the various algorithm versions and the estimated eye heights and widths, although the comparison of such speeds requires some care: As mentioned in Section 3.3.1, the numerical algorithm is time-agnostic (in the sense that it considers only the number of algorithm iterations, due to the probabilistic approach), therefore convergence speed has to be estimated by multiplying the number of iterations before convergence with the actual time corresponding to one iteration in the real implementation, i.e. $N_{\text{mean}}N_{\text{des}}T_{\text{B}}$. However, this is likely to result in an overestimation of the real convergence time, chiefly due to the fact that for each iteration the effect of all the possible sequences sent by the transmitter is considered (as explained in Section 3.3.1), instead of randomly using $N_{\text{mean}}N_{\text{des}}$ of such sequences; moreover, the numerical algorithm does not consider the 8b10b encoding applied at the transmitter, which reduces the number of possible sequences of a certain length by avoiding those likely resulting in the most severe ISI due to the large amount of consecutive equal bits. Regarding the other two algorithms, convergence speed is highly affected by the initial position of the sampling phase, which is not easily controllable: Starting close to the

data transitions (as happened for the behavioural algorithm with the PCI channel) intuitively requires a longer adaptation time to reach the optimal point, while an initial sampling edge in proximity of the eye centre (as occurred for the RTL implementation with the same channel) needs small adjustments only; with the test channel attenuating 14.3 dB, instead, the behavioural version of the algorithm started farther off the data transitions, thus almost halving the convergence time.

In general, Table 4.5 shows that the equalization parameters resulting from adaptation of the numerical and the RTL algorithms are very similar in both the cases considered; the behavioural algorithm, instead, tends to overestimate the $dLev$ and the DFE taps, even though taking into account the larger eye amplitude obtained in such a case results in comparable values with respect to the other two algorithm versions. The reported sampling phase shifts are all very similar (a difference of 0.02 UI is equal to less than 2 ps), due to the fact that small phase variations result in relevantly large reductions in the pre-cursor without affecting the main cursor too much. The eye heights for the various algorithm versions show more or less the same relations noted in the values of the adaptation parameters above, while the eye widths are very similar in all cases; the only exception is represented by the RTL algorithm in the PCI channel case, where both the eye height and width are rather lower than those obtained with the other algorithms. As mentioned above, care must be taken when analysing the convergence times: Due to the aforementioned reasons, the numerical results are likely to represent an overestimation, while the other two versions are highly dependent on the initial sampling phase; moreover, due to the digital implementation of the pulse response, which does not consider propagation time, different channels do not affect the initial sampling position (this is why the convergence times for the RTL version are very similar with both channels, as the final phase values are similar in magnitude), while the Verilog-A channels used in the transistor-level simulations properly incorporate delays so that the initial sampling phase varies with different channels.

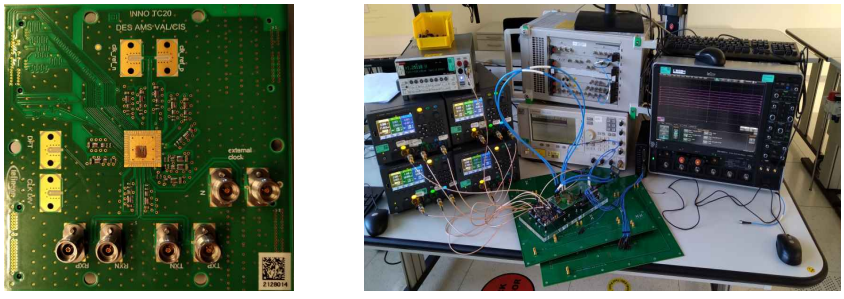
Chapter 5

Characterisation of the Test Chip Featuring Fully-Adaptive Equalization

This chapter reports on the characterisation of the new test chip (TC20) featuring the fully-adaptive equalization system presented in Chapter 4, which was based on a previous design (called TC18) upgraded to support the new features and improve performance, although targeting the same nominal 12 Gb/s data rate.

The characterization was performed at Infineon Technologies Villach (Austria) with the measurement setup shown in Figure 5.1b and consisting of the main PCB on which the bare-die chip is mounted (see Figure 5.1a), auxiliary PCBs to handle supply routing, while other ones with various test channels (some described in Section 4.1.1) can be connected when needed, power supplies, an Agilent Technologies E8257C analog signal generator as external clock source and a Teledyne Lecroy SDA 830Zi-B 30 GHz Serial Data Analyzers. An FTDI module connected to a PC is used to program the chip via JTAG interface.

For the following characterisation, three channels were considered: A stripline of 10 cm and two microstrips measuring 30 cm and 60 cm in length, respectively (the pulse response of TC18 for the 10 cm stripline at 11 Gb/s and for the 60 cm microstrip at 6.25 Gb/s are reported in Figure 4.3a).



(a) (b)

Figure 5.1: Panel (a) shows the PCB used for testing, while (b) shows the measurement setup connected in loopback mode.

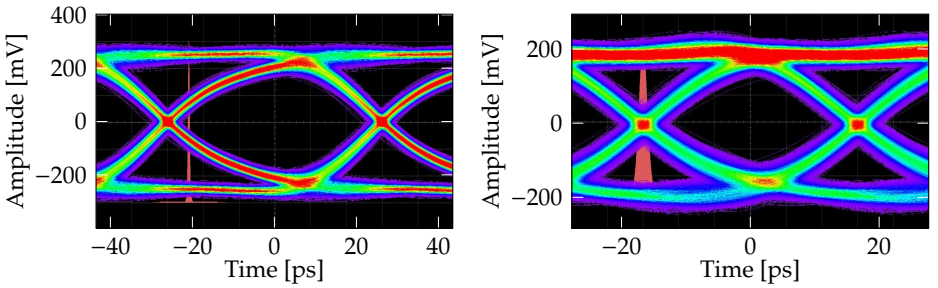


Figure 5.2: Measured eye diagram at the output of the transmitter at (a) 11.5 Gb/s and (b) 18 Gb/s without FFE; 8 driver replicas are activated, the supply voltage is V_{DD} = 0.9 V and a PRBS31 is transmitted. Despite it is evident that the serializer fails in (b), this plot shows that even at such a data rate (much higher than the nominal one) transmission is not limited by the circuit's bandwidth.

5.1 Characterisation of the Transmitter

The transmitter was characterised by observing the eye diagram at its output with the SDA 830Zi-B 30 GHz in various conditions, as shown in Figure 4.7; these only include the transmitter's breakout channel and 30 cm-long cables, the latter providing an attenuation of less than 2 dB. Plot (a) reports the eye diagram at 11.5 Gb/s without FFE, with V_{DD} = 0.9 V, 8 active driver replicas[1] and PRBS31, which features 330 mV eye height (580 mV voltage swing) and 76.6 ps (0.88 UI) eye width. Figure 5.2b is characterised by 210 mV eye height (450 mV voltage swing) and 43.3 ps (0.78 UI) eye width and was taken at the same conditions of plot (a) although at the higher data rate of 18 Gb/s.[2] The latter is much higher than the nominal value specified in the design, but provides useful information for further improvements: The upper, dark red band in the eye diagram, which signifies a non-equal proportion of '1' and '0' bits, entails that the serializer fails in correctly producing the data stream for the FFE switch matrix and the subsequent circuits; however, the fact that this results only in an unbalance between the transmitted high and low values means that the transmitter is not limited by any of the driver's or pre-driver's bandwidth, nor by the FFE switch matrix's timing (as one might expect from the apparently low margin of t_{LSB} in Figure 4.8a) because its effect would resemble that of an FFE with a post-tap activated (having a variable weight dependent on the number of resampling stages that do not satisfy the timing margins, as can be seen from the different paths and timing in Figure 4.7 of the 8 FFE slices controlled by the switch matrix with respect to those constantly driven with the main tap).

Eye diagrams were also used to assess the effect of adding a channel after the transmitter and observe the signal that is fed into the receiver when operating in

[1]Unless specified otherwise, 8 driver replicas are always used in the following to yield the nominal 50 Ω driver's output impedance; additional replicas may be enabled to increase the output swing at the expense of worse impedance matching.

[2]The only difference between the two cases concerns the power of the external clock fed to the transceiver, which had to be incremented when using higher frequencies to overcome limitations of the laboratory test bench: The clock was generated with 11 dBm at 11.5 GHz and with 28 dBm at 18 GHz.

Channel Length	10 cm	30 cm		60 cm
Data Rate	6.25 Gb/s	6.25 Gb/s	8 Gb/s	6.25 Gb/s
Transmitter				
$w_{\text{-}1}$	−1/16	−1/16	−2/16	−2/16
w_1	0	−1/16	−2/16	−1/16
Driver Replicas	12	8	12	12
Nominal Swing [V]	1.2	1.05	1.35	1.35
Channel				
$\lvert S_{21}\rvert$[a] [dB]	−3.7	−6.5	−10.3	−11.7
$\lvert S_{11}\rvert$[a] [dB]	−7.3	−11.5	−9.2	−8.9
Eye Height[b] [mV]	245	190	140	100
$V^{\text{b}}_{\text{swing}}$ [mV]	700	600	470	430
Eye Width[b] [UI]	0.75	0.67	0.54	0.50
Receiver				
VGA Gain[a] [dB]	6	4	6	6
CTLE Gain[c] [dB]	−2	−2	−2	−2
CTLE Peaking[a] [dB]	7	0	7	7
DFE Gain[a] [dB]	−4	4	6	6
Width[d] [UI]	0.16	0.44	0.34	0.59

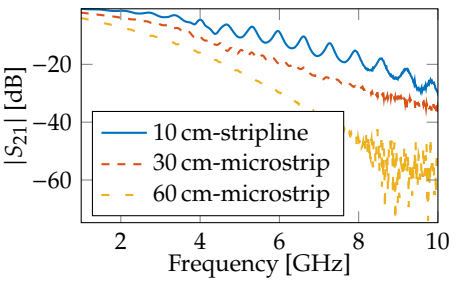
[a] At Nyquist Frequency.
[b] Refer to Figure 5.3c for such definitions.
[c] At DC.
[d] Estimated from the bathtub plot at BER= 10^{-12} after adaptive equalization.

Table 5.1: Transceiver settings used for characterising fully-adaptive equalization over lossy channels in the laboratory.

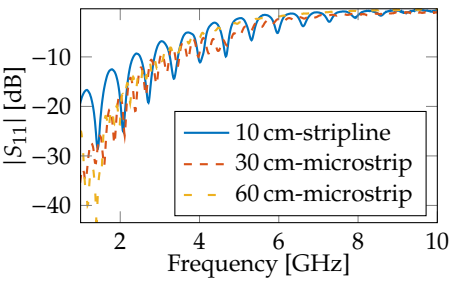
loopback mode (see the next Section). The three PCB channels (the 10 cm-long stripline and the two microstrips of 30 cm and 60 cm) were connected and the corresponding eye diagrams where acquired when the transmitter was set with the same configuration that will be used for the measurements reported in the next Section and presented in Table 5.1. Such are not optimal settings for best reception, instead they were chosen only to provide a good enough starting point (i.e. recognising the PRBS and resulting not in too a high BER) for the fully-adaptive algorithm to be able to work correctly.

Such eye diagrams were acquired with the SDA 830Zi-B 30 GHz and are shown in Figure 5.3.[3] With the 10 cm-stripline, the 6.25 Gb/s signal at the output of the channel is characterised by an eye height of 245 mV, a differential swing of 745 mV and an eye width of 120.2 ps (0.75 UI); transmitting at the same frequency over a microstrip with length of 30 cm yields 190 mV eye height, 665 mV differential swing and 106.8 ps eye width (0.67 UI), whereas with the 60 cm-microstrip height, swing and width become 100 mV, 500 mV and 79.7 ps (0.5 UI), respectively. The eye diagram at 8 Gb/s after the 30 cm-microstrip features an eye height of 140 mV, a differential swing of 470 mV and an eye width of 67.6 ps (0.54 UI).

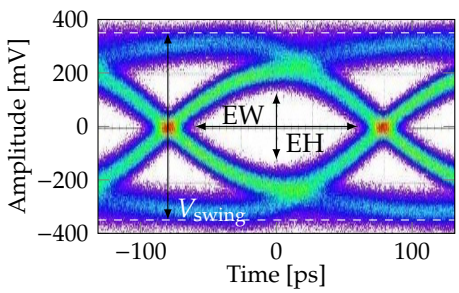
[3] Note that such eyes are obtained with different settings on the amount of acquired samples, as can be seen from the colours that represent trace density.



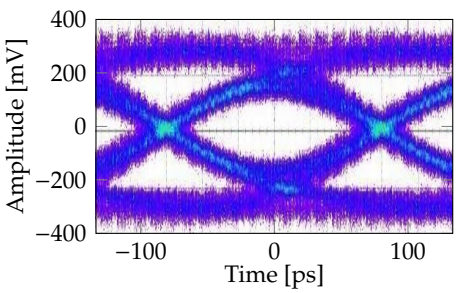
(a) Test channels: $|S_{21}|$



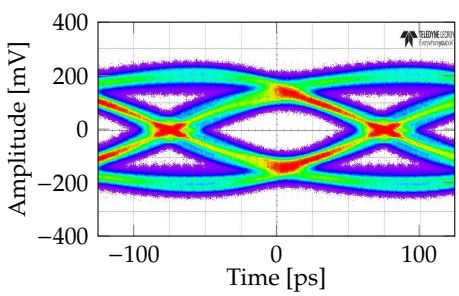
(b) Test channels: $|S_{11}|$



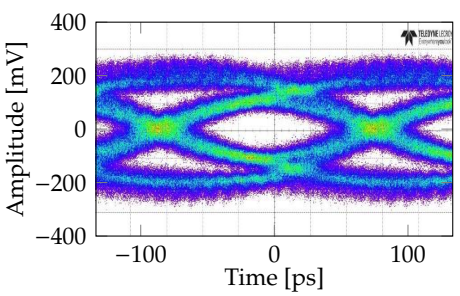
(c) 10 cm-stripline, 6.25 Gb/s



(d) 30 cm-microstrip, 6.25 Gb/s



(e) 30 cm-microstrip, 8 Gb/s



(f) 60 cm-microstrip, 6.25 Gb/s

Figure 5.3: Test channels' (a) $|S_{21}|$ and (b) $|S_{11}|$ parameters, while (c) to (f) report the measured eye diagrams at the output of the various channels with the transmitter settings of Table 5.1 when sending a PRBS31 with $V_{DD} = 0.9\,V$. The definition of Eye height EH, eye width EW and voltage swing V_{swing} are shown in plot (c) as a reference for the measurements reported in Table 5.1.

5.2 Loopback and Fully-Adaptive Equalization

The receiver and the fully-adaptive equalizer were tested in loopback mode, i.e. by connecting the transmitter's output to the receiver on the same chip (hence sharing the same clock), possibly placing channels from the dedicated board in between and measuring the resulting bathtub plots.

Fully-adaptive equalization over the 10 cm-long stripline losing 3.7 dB at the Nyquist frequency is shown in Figure 5.4 in the case of transmission of a PRBS31 at 6.25 Gb/s with the settings of Table 5.1. Plot (a) reports the adaptation of $dLev$ and of the three DFE taps a_n when the loop was run with $N_{\text{mean}} = 2^8$, $\mu_{\text{dLev}} = 2^{-3}$ and $\mu_{\text{DFE}} = 2^{-4}$ (such values will be used for all other test cases, unless otherwise specified), represented in Figure 4.22 by the shifter $\ll N$ between the accumulator and the DAC. Notice that the values of the adapted parameters, i.e. $dLev$, a_n or φ_{shift}, cannot be measured directly but the corresponding digital words can be read from the digital macro through a monitoring bus; therefore, when reporting the values of the corresponding physical quantities, the nominal LSB was considered (i.e. 7 mV for $dLev$, 2.5 mV for a_n and UI/32 for φ_{shift}, determined respectively by the simulated characteristics of the variable threshold comparator, of the IDAC and of the PI, all presented in Section 4.2.2) to convert the digital codes, and average was performed to correctly present the effective values after convergence, which typically toggle among few different codes. Due to the limited width of the digital monitor bus used to acquire the adapted parameters from the digital macro, only one quantity at the time could be observed, which does not allow to determine the actual start time of the adaptive algorithm but only the instant when the observed quantity begins to vary, thus triggering the SDA. This entails that, when observing adaptation as in Figure 5.4a, all parameters start moving at time 0 ms by convention, whereas the DFE taps are expected to begin adapting only after $dLev$ "reaches" the top band of the eye diagram (e.g. see plots (a) and (c) in Figure 3.11).

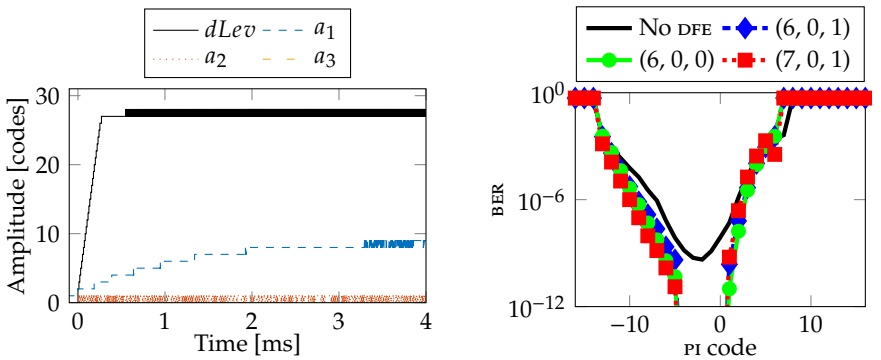


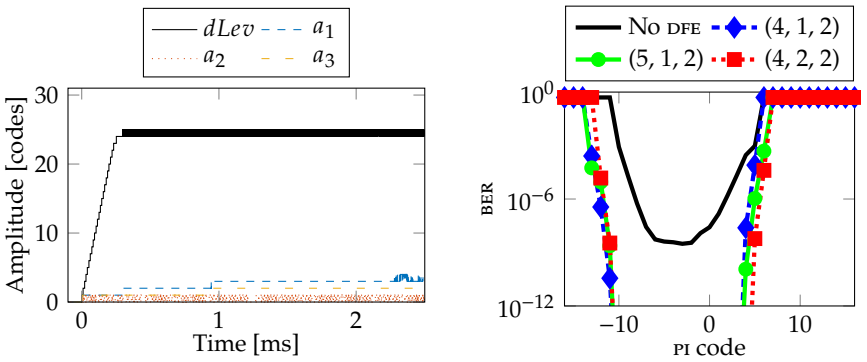
Figure 5.4: Fully-adaptive equalization at 6.25 Gb/s over the 10 cm-long microstrip losing 3.7 dB at the Nyquist frequency; a PRBS31 is transmitted with the settings of Table 5.1. Plot (a) shows the adaptation over time of $dLev$ and of the DFE taps; plot (b) shows the bathtubs obtained without DFE (solid black line), with the DFE adapted by the digital algorithm (dashed blue line) and with the DFE programmed manually (remaining lines) for comparison.

A few remarks on the acquisition of the bathtubs have to be made before discussing the corresponding results. First of all, for all configurations (except the ones of Figures 5.7c and 5.7d) the resulting bathtubs are centred according to the CDR code measured *without* enabling the DFE, so that the effect of such a correction on the post-cursors is not taken into account by the CDR. Also note that the bathtubs have to be acquired with fixed DFE settings (i.e. without the adaptive loop running), otherwise sweeping the PI code would cause the SS-LMS algorithm to use data and errors sampled in different points along the eye, hence not behaving as in a real situation. Moreover, the adaptive loop seldom converges to a single value for each parameter, rather the various $dLev$ and DFE taps toggle among a few values after convergence; therefore, when referring to the "adapted DFE taps" in the bathtub plots, this shall be interpreted as the most common triplet (a_1, a_2, a_3) occurring after convergence, whereas some of the configurations set manually (and shown in the same figure) may also be comprised in the values covered by the SS-LMS algorithm.

Another relevant remark regarding the bathtubs and the plots that report the adaptation of $dLev$ and of the taps a_n (diplayed side by side in the following) is that the DFE settings used for the bathtubs do not necessarily correspond to the taps value after convergence, e.g. the bathtubs in Figure 5.4 were acquired with $a_1 = 6$–7, whereas in the plot on the left the first tap converges between 8 and 9. Such a regrettable situation is consequence of the limited time available for characterisation and the way it was performed from remote: Briefly speaking, at first the fully-adaptive algorithm was run and the values of the DFE taps were read through JTAG only after convergence (i.e. without the possibility to look at the whole adaptation transient); these were then used to acquire the bathtubs. Only in a different moment (e.g. at least after the lengthy acquisition of all the bathtubs to a BER bottom line of 10^{-12}, sometimes even *days* earlier or later) the SS-LMS loop was run again and the whole adaptation transient of $dLev$ and a_n was stored from the digital monitor and processed for visualization, due to the required presence of personnel in the laboratory to perform the acquisition; environmental conditions may have changed in the meanwhile, affecting the repeatability of the measurement and thus altering the correspondence between the different runs of the adaptive algorithm, and the lack of time did not allow measurements to be re-performed properly.

Figure 5.4b reports the bathtubs that, without DFE, do not even reach a BER of 10^{-10}, whereas after adaptation has converged to $dLev = 187.1\,\text{mV}$, $a_1 = 18.5\,\text{mV}$, $a_2 = 1.6\,\text{mV}$ and $a_3 = 0\,\text{mV}$ the aperture greatly increases and the bottom line reaches a BER of 10^{-12}; moreover, by manually programming the DFE taps, no configuration was found that could outperform the one obtained through the adaptive algorithm, thus demonstrating its effectiveness, whereas other values covered by the SS-LMS loop when oscillating after convergence (shown in the same plot) result in comparable bathtubs.

Figure 5.5 shows the effect of fully-adaptive equalization performed in the case of transmission of a PRBS31 at 6.25 Gb/s over the 30 cm-long microstrip losing 6.5 dB at the Nyquist frequency, with the settings of Table 5.1. Plot (b) shows that, without DFE, the centre of the eye cannot even reach a BER of 10^{-9}, whereas after adaptation has converged to $dLev = 165.2\,\text{mV}$, $a_1 = 6.8\,\text{mV}$, $a_2 = 1.4\,\text{mV}$ and $a_3 = 2.5\,\text{mV}$ the aperture greatly increases; other DFE taps configurations (either comprised in the convergence of the adaptive loop or set manually) either result in comparable or lower performance.



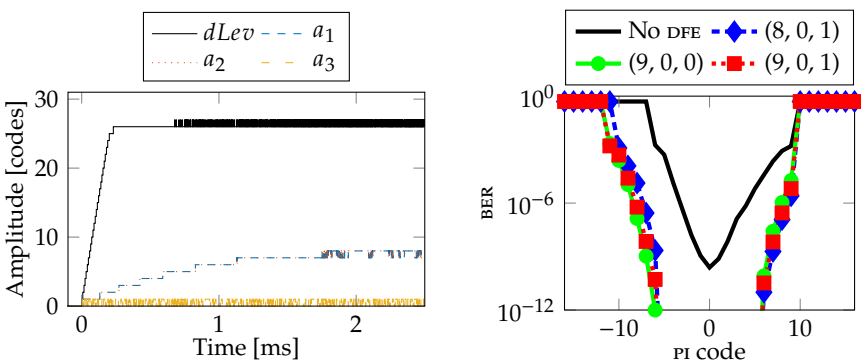
(a) (b)

Figure 5.5: Fully-adaptive equalization at 6.25 Gb/s over the 30 cm-long microstrip losing 6.5 dB at the Nyquist frequency; a PRBS31 is transmitted with the settings of Table 5.1.Plot (a) shows the adaptation over time of $dLev$ and of the DFE taps; plot (b) shows the bathtubs obtained without DFE (solid black line), with the DFE adapted by the digital algorithm (dashed blue line) and with the DFE programmed manually (remaining lines) for comparison.

Fully-adaptive equalization was tested on the same channel at 8 Gb/s (−10.3 dB at the Nyquist frequency) with a PRBS31 and the settings of Table 5.1. Adaptation converges to $dLev$ = 176.1 mV and DFE taps of a_1 = 15.4 mV and a_3 = 1.5 mV.[4]

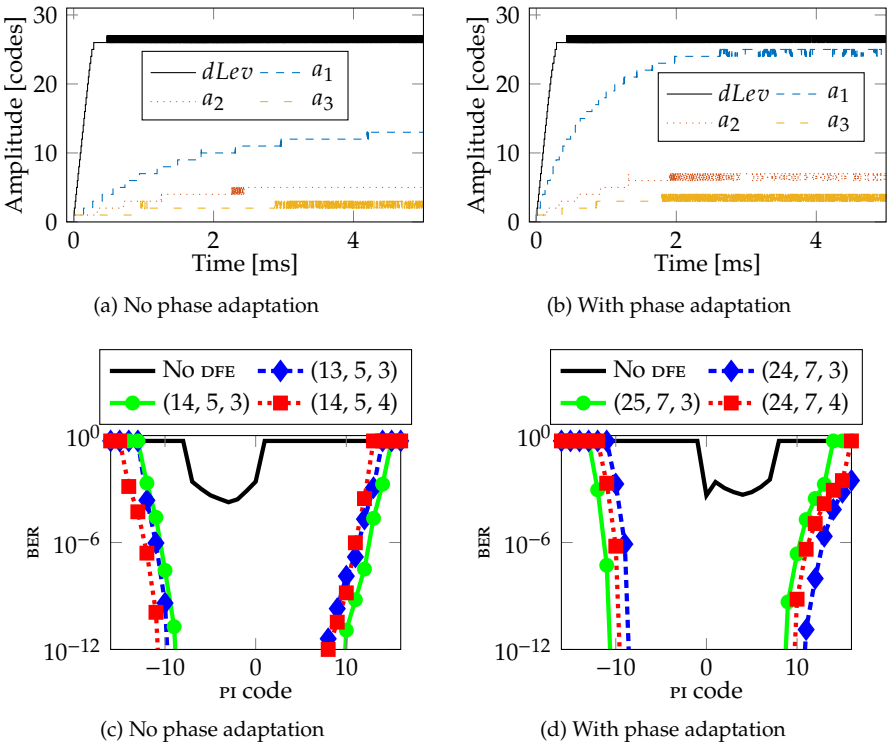
A final test for the SS-LMS algorithm is to check full adaptation of the sampling

[4] a_2 could not be determined due to an error in the acquisition.



(a) (b)

Figure 5.6: Fully-adaptive equalization at 8 Gb/s over the 30 cm-long microstrip losing 10.3 dB at the Nyquist frequency; a PRBS31 is transmitted with the settings of Table 5.1. Plot (a) shows the adaptation over time of $dLev$ and of the DFE taps; plot (b) shows the bathtubs obtained without DFE (solid black line), with the DFE adapted by the digital algorithm (dashed blue line) and with the DFE programmed manually (remaining lines) for comparison.



(a) No phase adaptation

(b) With phase adaptation

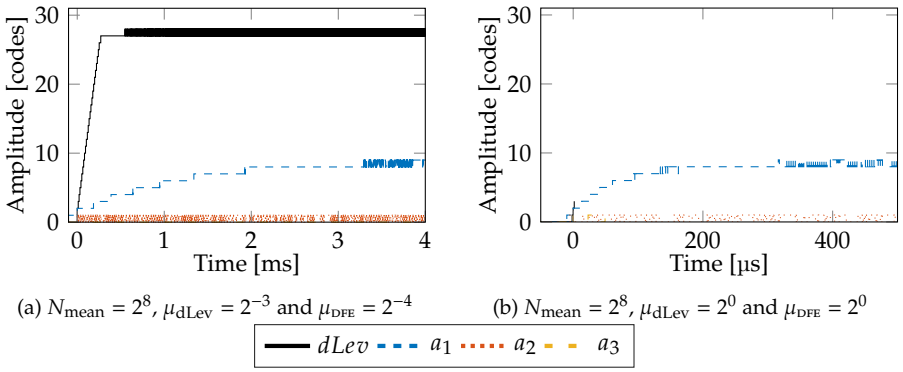
(c) No phase adaptation

(d) With phase adaptation

Figure 5.7: Fully-adaptive equalization at 6.25 Gb/s over a 60 cm-long microstrip losing 11.7 dB at the Nyquist frequency; a PRBS31 is transmitted with the settings of Table 5.1. Plot (a) shows the adaptation over time of $dLev$ and of the DFE taps; plot (b) shows similar results but after optimal sampling phase adaptation; plots (c) and (d) show the bathtubs obtained without DFE, with the DFE adapted by the digital algorithm and with the DFE programmed manually for comparison for the cases of plots (a) and (b).

phase, thus aiding the CDR in finding the optimal sampling point in the eye diagram. This was verified in the case of transmission of a PRBS31 at 6.25 Gb/s over the 60 cm-long microstrip losing 11.7 dB at the Nyquist frequency, with the settings of Table 5.1. Plot (c) and (d) show the bathtubs without and with sampling phase adaptation: In the former case $dLev$ converges to 181.3 mV and the DFE taps to $a_1 = 23.1$ mV, $a_2 = 8.6$ mV and $a_3 = 5.9$ mV, whereas after the sampling phase has adapted, resulting in a time offset (or φ_{shift}) corresponding to -5 PI codes (-12.5 ps, or -0.156 UI), the SS-LMS loop reaches $dLev = 182.9$ mV, $a_1 = 54.1$ mV, $a_2 = 15.5$ mV and $a_3 = 8.7$ mV. This is in line with the simulation results presented in Section 3.3.2, where the adaptation of the sampling phase was shown to modify both $dLev$ and the DFE taps, particularly concerning the increase of the DFE weights due to the earlier sampling phase, which results in larger first few ISI post-cursors. The fact that the digital monitor outputs φ_{shift} already added to the control word coming from the CDR did not allow to visualise the adaptation of the optimal sampling phase.

The 10 cm-stripline channel with 6.25 Gb/s transmission and the settings of



(a) $N_{\text{mean}} = 2^8$, $\mu_{\text{dLev}} = 2^{-3}$ and $\mu_{\text{DFE}} = 2^{-4}$

(b) $N_{\text{mean}} = 2^8$, $\mu_{\text{dLev}} = 2^0$ and $\mu_{\text{DFE}} = 2^0$

(c) $N_{\text{mean}} = 2^2$, $\mu_{\text{dLev}} = 2^{-3}$ and $\mu_{\text{DFE}} = 2^{-4}$

(d) $N_{\text{mean}} = 2^9$, $\mu_{\text{dLev}} = 2^{-3}$ and $\mu_{\text{DFE}} = 2^{-4}$

Figure 5.8: Fully-adaptive equalization at 6.25 Gb/s over a 10 cm-long microstrip losing 3.7 dB at the Nyquist frequency; a PRBS31 is transmitted with the settings of Table 5.1. Plot (a) to (d) show the adaptation over time of $dLev$ and of the DFE taps for various values of the parameters N_{mean}, μ_{dLev} and μ_{DFE} (such results are summarised in Table 5.2).

Table 5.1 was used to test adaptation with varying algorithm parameters, chiefly N_{mean}, μ_{dLev} and μ_{DFE}. Adaptation of the various parameters is shown in Figure 5.4a (notice the different time scales), whereas Table 5.2 summarises such results. The effect on the bathtubs caused by the different DFE taps values after convergence in the considered cases could not be investigated due to the limited time available for characterisation.

The power consumption of the different blocks can be identified thanks to the presence of various supply domains, and the corresponding power efficiencies are reported in Table 5.3 in the loopback case without a channel, activating or not the adaptive loop running. The main responsible for power consumption is the RX analog part (considering that power due to the clock distribution network is under the transmitter's supply domain) and the increased contribution due to operation of the SS-LMS algorithm (chiefly due to the required additional slicers, deserialisers, PI etc., as reported in Section 4.2.2) is not negligible, accounting for a 17 % higher consumption when enabled.

Table 5.4 compares the transceiver featuring fully-adaptive equalization characterised in this Chapter with other transceivers operating at about the same data rate and featuring adaptation of their equalizers.

N_{mean}	μ_{dLev}	μ_{DFE}	$dLev$ [mV]	a_1 [mV]	a_2 [mV]	a_3 [mV]	$t^{\text{a}}_{\text{conv}}$ [ms]
2^8	2^{-3}	2^{-4}	187.1	18.5	1.6	0	3.3
2^8	2^0	2^0	N.A.[b]	21.3	1.6	2.5	0.4
2^2	2^{-3}	2^{-4}	182.6	15.3	0	5.3	0.04
2^9	2^{-3}	2^{-4}	174.7	10.6	0	2.8	4.8

[a]Convergence time estimated from Figure 5.8, hence assuming the DFE taps start adapting at time 0.
[b]Due to an error in the acquisition.

Table 5.2: Convergence values and time of the adaptation parameters shown in Figure 5.8 and corresponding to the plots of Figure 5.4.

Adaptive	Power Efficiency [mW/(Gb/s)] Off	On
Transmitter	5.78	5.76
Driver + Pre-driver	2.51	2.50
Serializer + FFE + Resampling	3.27	3.26
Receiver	3.54	4.90
VGA + CTLE + DFE + Slicer	1.05	2.45
PI + Deserializer	2.49	2.45
Digital	0.55	0.62
Clocking and biasing	2.88	2.85
Total	12.75	14.13

Table 5.3: Power consumption normalised to the bit rate at 6.25 Gb/s without and with fully-adaptive equalization enabled for $V_{\text{DD}} = 0.9$ V. Main PCB + cables.

	This work	[20]	[91]	[123]
Node [nm]	28	65/180[a]	28	40
Speed [Gb/s]	6.25	6	6.6	8
Signalling scheme	PAM-2	PAM-2	PAM-2	PAM-4
FFE taps	3	3	3	3
DFE taps	3	5	0	2
Adaptation	DFE, φ_{opt}	DFE	CTLE	VGA, CTLE, DFE
Channel loss [dB]	11.7	24	20	22
Power [mW/(Gb/s)]	14.13	n.a.	19.5	8.15
V_{DD} [V]	0.9	n.a.	1, 1.2	1.1
Area [mm^2]	0.11[b]	4.1[c]	0.64[b]	0.59[d]

[a]TX/RX
Layout areas include: [b]TX +RX, [c]4TX +4RX +PLL, [d]TX +RX +PLL

Table 5.4: Transceiver performance at 6.25 Gb/s and comparison with similar works.

Chapter 6

Design and Simulation of a 16 Gb/s Wireline Transmitter for Automotive Applications

6.1 Transmitter Design

The target of the activity covered in this chapter was to design a 16 Gb/s transmitter in a planar 28 nm technology node, supplied by $V_{DD} = 0.9$ V and providing a 0.9 V peak-to-peak differential output swing. The transmitter should feature reduced tap-range FFE (two or three taps) with a decent 5 b granularity, thus providing correction steps of ≈ 0.56 dB. Such a design poses many challenges both at the system/architectural level and at the circuit level:

- The rate of the architecture (either full or half rate, as briefly mentioned in Section 1.3.4) becomes a critical choice in determining power consumption, requirements for clock distribution and timing constraints in the serializer;
- the transmitter paradigm may be re-considered, possibly moving to an architecture based on *digital-to-analog converters* for enhanced versatility (i.e. employing different signalling schemes and equalization architectures);
- the driver bandwidth may become critical and hinder reaching high data rate;
- switching noise increases and can negatively affect the transmitter's jitter performance;

Any of the challenges above comes with its own trade-offs, so that a choice can be made only after careful evaluation of the system specifications and of the interactions with other blocks.

6.1.1 General Half-Rate Architecture

For what the transmitter is concerned, the main challenge linked to the target data rate is to ensure that the serializer can safely operate under all PVT conditions, which means that the timing margins must be respected at all times. If this is not the case, the parallel data provided to the transmitter would be converted into a serial stream having (at least some) bits ordered in the wrong sequence, entailing an increase in BER at the receiver. Satisfying all timing constraints with a full-rate transmitter entails that the critical path corresponding to the final multiplexing stage (composed

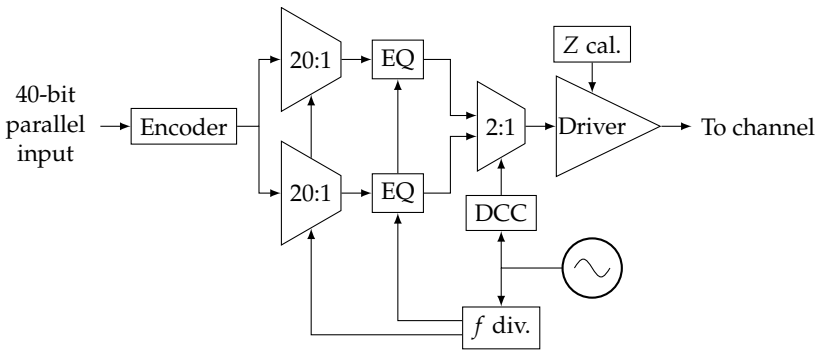


Figure 6.1: Simplified block diagram of the half-rate transmitter. DCC is the duty cycle-correction circuit, EQ is the FFE switch matrix and "Z cal." is the calibration circuit for the driver's impedance; the clock source is external.

of a flip-flop's clock-to-output and setup time, plus propagation time of a MUX, see Figure 1.21) must be shorter than 62.5 ps, whereas the same constraint in a half-rate architecture is applied to a shorter path (composed only of a flip-flop's clock-to-q time, plus propagation time of a MUX, as shown in Figure 1.21) [25]. Moreover, removing the final flip-flop means less components operating at full rate (16 Gb/s), thus reducing power consumption and high-frequency switching noise.

One major advantage of moving to a half-rate architecture is that the whole transmitter can be clocked at 8 GHz instead of 16 GHz, hence allowing relevant savings in power consumption and less challenges in clock synthesis and distribution due to the lower clock frequency involved [25]. However, the fact that both the clock's levels are used to multiplex the data directly into the driver without retiming (compare the full- and half-rate transmitters in Figure 1.21) makes such an architecture sensitive to *duty cycle distortion (DCD)*, which would make even and odd bits of different lengths, effectively closing the eye at the receiver because the CDR cannot track such a jitter source; this mandates a *duty-cycle correction circuit* at transmit side to restore a clock with 50 % duty cycle [25]. Even when generating a full-rate clock to be divided locally (despite having to handle clock distribution at higher frequency), perfect 50 % duty cycle cannot be guaranteed due the inevitable remaining distribution circuitry after frequency division and load of further blocks that require the half-rate clock.

Despite the aforementioned challenges associated with the half-rate architecture, this was deemed a promising solution for increasing the data rate without facing issues linked to clock generation and distribution, switching noise (and consequently PSIJ, see Appendix A.4) and timing constraints.

A simplified block diagram of the transmitter designed in this Chapter is reported in Figure 6.1: The 40-bit parallel input is encoded inside the digital macro and then fed to the analog portion of the circuit, where it is serialised and equalized before being multiplexed into a full-rate, 16 Gb/s data stream and then driven into the channel. The half-rate 8 GHz clock is generated outside the chip, it goes through a duty cycle-correction circuit to time the final multiplexer and it is divided multiple times to clock the serializer and FFE; an impedance calibration circuit ensures that the driver's output impedance matches the characteristic impedance of the channel under all operating conditions.

6.1.2 Conventional vs. DAC-Based Transmitter

The choice of the driver required to investigate first the possibility to shift to a DAC-based transmitter, implementing a digital-to-analog converter instead of a conventional driver, so that FFE compensation can be set directly by the input amplitude code to the DAC, thus removing the shift register and switch matrix typically implemented to feed the driver with the required FFE-weighted data , avoiding altogether segmented drivers (to impose the FFE weight to the transmitted pulse) and the associated output parasitics that hinder performance [124].[1] This has the main advantage of enabling versatile transmit-side equalization and signalling, because computation of FFE correction and implementation of multi-level signalling schemes are taken care of by the digital macro, thus allowing dynamic assignment of equalisation taps and multi-mode signalling on the same transmitter architecture, assuming that enough DAC resolution is provided [125].

Moving to a DAC-based transmitter implies a shift of complexity and focus from analog/mixed-signal to digital design, where many critical operations are now dealt with. Particularly, efficient FFE implementation is beneficial from both a computational and an energetic standpoint: In fact, reducing the computational burden is required to reduce area-consuming hardware resources, satisfy latency requirements for providing the FFE-weighted data and keep power consumption at acceptable levels. This is typically achieved by implementing *look-up tables* (LUTS) that store pre-computed FFE weights corresponding to data sequences of a certain length (determined by equalization requirements in terms of tap range); such sequences simply serve as addresses to retrieve the corresponding codes from the LUT and send them to the analog transmitter block [124, 125].

Such flexibility comes with drawbacks specific to design of the DAC (e.g. linearity, improving which may require implementing both binary and thermometer DAC bits [125,126]), power consumption of the digital backend due to the DSP equalizer [125] and area (and, yet again, power) overhead in the analog/mixed-signal blocks, since each bit that composes the DAC's input amplitude code requires its own serializer and retiming [125] (i.e., with respect to a conventional transmitter, the data path has to be replicated by the number of bits required to achieve the desired DAC resolution). Although challenges related to the DAC can be tackled with proper design techniques, the latter issues may actually restrict implementation of DAC-based transmitters to applications where power consumption is not the most relevant metric, being speed the main requirement.

In order to decide whether to implement a conventional or a DAC-based half-rate transmitter, the two architectures shown in Figure 6.2 were designed and compared. As a proof of concept, the DAC implements 4 bits and 8 driver replicas for impedance calibration (refer to Section 6.1.6 for further details), while the conventional transmitter features 15 segments for the FFE weights for the sake of a fair comparison (a DAC's 4 bits are implemented as 15 binary-weighted segments) and 8 driver replicas to set the output impedance. Both of the implementations feature source-series terminated, voltage-mode p-over-n drivers, which were preferred to current-mode drivers due to their intrinsic reduced power consumption [25].

In the implementation of Figure 6.2a, the PRBS generator provides four 40-bit wide buses, hence binary encoding the data sequence which is then fed into four

[1] Strictly speaking, the DAC is also composed of segments, which are hardwired so to represent different bit weights (see Figure 6.2), whereas in a conventional driver the segments can be dinamically clustered so to provide different FFE weights.

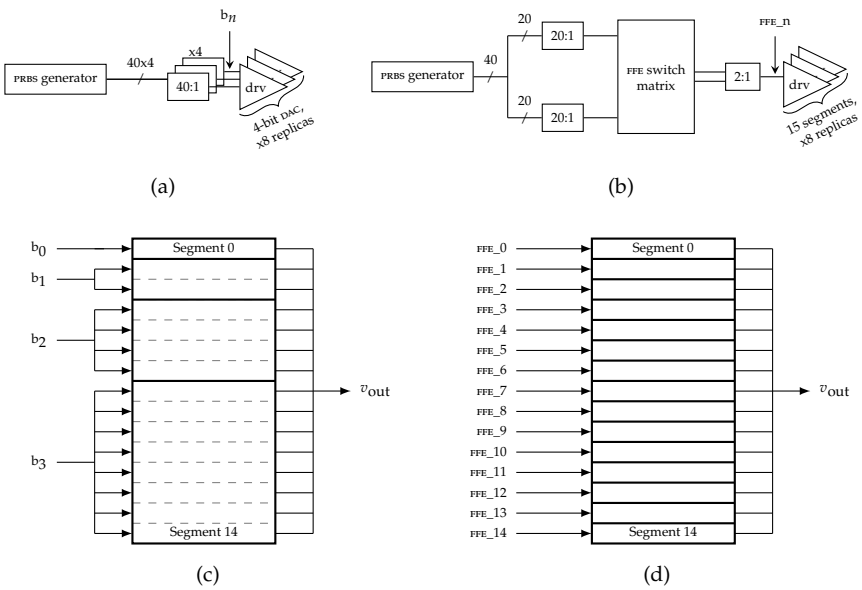


Figure 6.2: Block diagrams of (a) 4-bit DAC-based and (b) 15-segments conventional transmitter, as implemented for comparing the two architectures; all signals are shown as single-ended for simplicity. Both of them feature 8 driver replicas for impedance calibration; the DSP performing FFE is omitted in (a) to restrict simulations to the analog/mixed-signal circuit (equalization would be performed immediately after the PRBS generation, in the digital domain), while the FFE in (b) can implement two or three taps. Details of the difference in connections between the DAC and the conventional driver are shown in (c) and (d), respectively: b_n are the bits encoding the (possibly equalised) data bit voltage level after the 40:1 serializer in (a), whereas FFE_n are the outputs of the FFE switch matrix after the final 2:1 serializer stage in (b).

replicas of the 40:1 half-rate serializer before reaching the DAC at full rate (when actually designing a DAC-based transmitter, FFE is applied directly at the output of the PRBS generator, so that the buses that feed the serializers include the FFE correction). Instead, in the conventional architecture shown in Figure 6.2b the PRBS generator provides a single 40-bit wide bus which feeds the 40:1 half-rate serializer with embedded FFE; this mandates splitting the serializer in even and odd data paths (as explained in greater detail in Section 6.1.4), working with full-rate data only after the FFE switch matrix. Two instances of the FFE in the conventional architecture are implemented to feature two and three taps, respectively, in order to explore also this portion of the design space, although such results would not be immediately comparable with the DSP, since it was not implemented at this stage. Notice that in the conventional architecture the FFE switch matrix not only connects the driver's segments to the correct delay stage, it also applies a specific sign to the various taps in order to implement negative weights, whereas in the DAC-based architecture the different signs of the FFE taps are accounted for in the digital macro when generating the control words to be sent to the serializer.[2]

[2]Consider that, employing a differential signalling scheme, each driver segment can contribute to

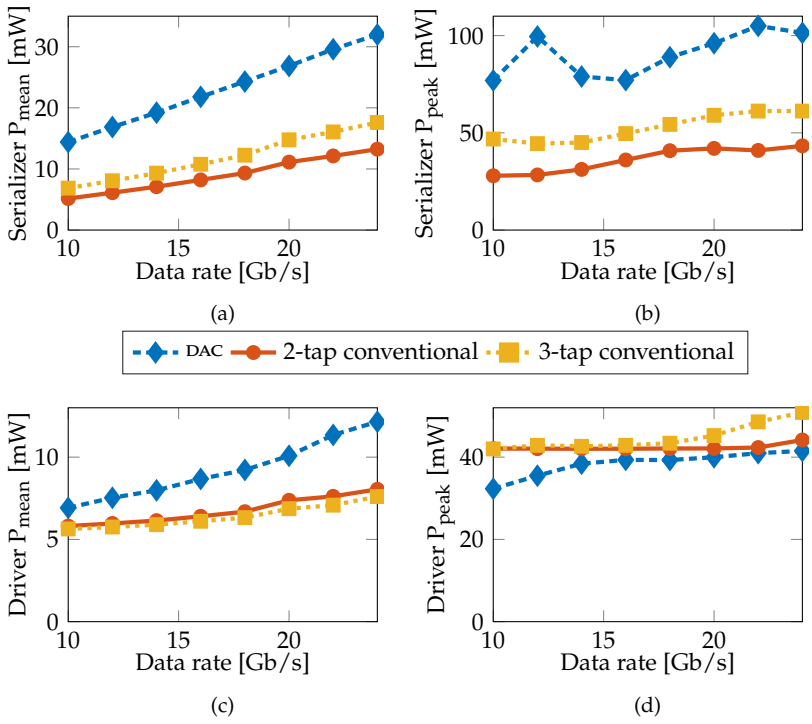


Figure 6.3: Simulation of a DAC-based transmitter and a conventional 2- and 3-tap FFE half-rate transmitter at various data rates to compare the serializers' (a) mean and (b) peak power consumption, and the drivers' (c) mean and (d) peak power consumption. Simulations were run with PRBS31 and supply voltage of 1 V.

The two architectures were simulated with 1 V supply voltage at various data rates from 10 Gb/s to 24 Gb/s and power consumption was measured separately for the serializer and the driver in order to determine possible relevant trends. The results are reported in Figure 6.3 and show mean and peak power for the two blocks, the latter having been chosen as a proxy for noise induced on the supply network that can couple back to other circuits. Figures 6.3a and 6.3b clearly highlight the fact that the replicated serializer required to drive the DAC is more power hungry at all data rates, although it is evident that such a trend heavily depends on the resolution of the DAC (the number of serializer instances equal that of the DAC's bits) and on the number of FFE taps implemented in the conventional architectures (more FFE taps require larger switch matrices and more retiming stages): In fact, it looks as though implementing longer FFE tap ranges would render the DAC-based architecture more convenient despite the overhead in the serializer architecture; however, such a result also depends on the location of the FFE, because power might be reduced if it were implemented earlier in the serialization flow, albeit at the cost of larger area required for the subsequent portion of the serializer, because it has to be replicated depending on the granularity of the FFE taps (see Section 6.1.4).

Figures 6.3c and 6.3d report the driver's power consumption, although the

either positive or negative output voltages, depending on its input bit being either '1' or '0'.

trends may seem contrasting the fact that the two implementations were topologically identical (see Figure 6.2): Even though they featured the same number of instances (15 segments and 8 driver replicas), the serializers architecture imposed different pre-driver stages, which were not optimised at the time of design space exploration (neither taken into account in the measured power consumptions) and resulted in different rise and fall times of the driver's input between the DAC and the conventional architecture. This means that the slower transitions that occur in the DAC due to the non-optimal pre-drivers cause the average power consumption to rise due to crowbar currents, in turn reducing its power peaks.

Summarising, although the DSP power consumption was not taken into account in the analysis above[3], it looks as though the optimal point where to switch from a conventional transmitter architecture to a DAC-based one depends mainly on the transmit-side equalization and the DAC resolution required to meet the application specifications. Given that the targeted transmitter required 3 FFE taps at most and that at least 5 bits were deemed necessary to achieve the desired FFE resolution[4], the versatility of a DAC-based transmitter was traded for the power savings that come from a low-range FFE in a conventional architecture.

6.1.3 Driver

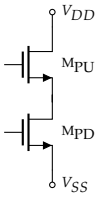
As mentioned in the Section above, the driver was chosen to be a voltage-mode (VM) topology, mainly for its low power consumption and high linearity compared to a current-mode (CM) driver. This implies that an impedance calibration scheme has to be devised to tune the driver's output impedance in order to match the characteristic impedance of the channel and thus eliminate reflections (refer to Section 6.1.6), whereas the impedance of a CM driver is simply set by its load resistors. Moreover, specific blocks to produce FFE-weighted data from the serial bit stream have to be added to the transmitter (instead of simply modifying the tail currents in a CM driver); additional drawbacks of VM drivers comprise a variable output impedance during switching and limited *power-supply rejection ratio* (PSRR) [15, 25, 108, 124, 127, 128].

A consequent choice regards the implementation of the driver segments as *n-over-n* or *p-over-n*, as depicted in Figure 6.4. Relevant features of p-over-n drivers include high output swings (which makes them typically preferred in applications that face severe channel losses) and lower output impedance variation during switching, although their power consumption has to be traded off with pre-driver strength, jitter and timing characteristics [25, 108].

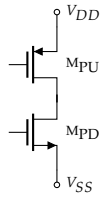
Transmitters with n-over-n drivers are favourable in terms of output impedance, since it can be set by the pre-driver's supply through the V_{GS} applied to the driver, although they differ for pull-up and pull-down NMOS (the top one's is even dependent on the data) [25, 108]. Targeting high-loss automotive channels, the p-over-n solution was deemed preferable to increase the signal swing, although an appropriate strategy for impedance calibration has to be implemented.

[3] It was supposed to be carried out in case the two architectures resulted fairly comparable, but eventually it would only have increased the power consumption gap of the the DAC option with respect to the conventional one.

[4] The previous version of the transmitter [76, 114], presented in Section 4.1.1, featured an FFE with a granularity of 4 bits, which appeared to be coarse, hence not very effective in equalizing channels with different ISI.

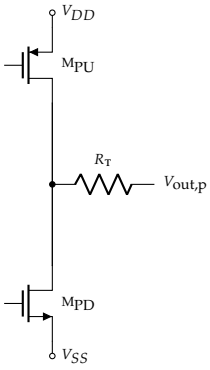


(a) Voltage-mode n-over-n driver.

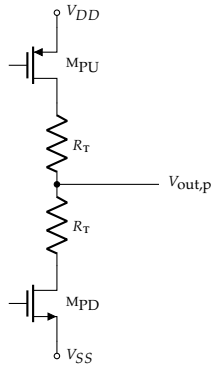


(b) Voltage-mode p-over-n driver.

Figure 6.4: The two alternative VM driver topologies.



(a) Series termination.



(b) Shunt termination.

Figure 6.5: The two most common termination strategies for VM p-over-n drivers.

Once the driver topology is chosen, the termination resistance R_T (refer to Section 1.3.2) has to be properly set and placed to match the channel's characteristic impedance when combined with the driver's own output impedance [25]. The two most common termination solutions (*series* and *shunt*) are shown in Figure 6.5: Series-terminated drivers are typically designed so that the resistor dominates the transistors' on impedance, thus the MOSS have to be rather large and might require stronger pre-drivers, although such transistors provide less variation in high-frequency output impedance [25, 39, 129]; moreover, direct connection between V_{DD} and V_{SS} when switching implies large crowbar currents and related supply integrity issues [25]. Shunt-terminated drivers reduce crowbar currents by shunting the channel to the MOSS' drains and can speed up switching if the transistors become a significant proportion of the termination impedance; this also means that they can be made smaller, hence reducing pre-driver loading due to gate capacitance [25, 126].

The actual design of the driver was performed by Hongyang Zhang from Infineon Technologies Villach, and is here reported to provide a complete description of the transmitter's circuits.

Such alternative termination strategies were tested in simulations to gain a better understanding of their pros and cons. At first, a test bench to verify their power consumption performance was set up by implementing a supply network with $R = 100\,\mathrm{m\Omega}$ and $C = 30\,\mathrm{pF}$, $L = 1.5\,\mathrm{nH}$ for V_{DD} and $L = 0.5\,\mathrm{nH}$ for V_{SS}

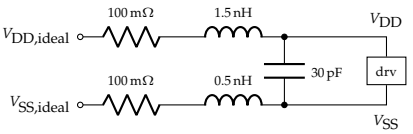


Figure 6.6: Power delivery network used to compare the two driver topologies.

(shown in Figure 6.6), with 300 fF driver's output capacitors mimicking the load due to typical *electro-static discharge* (ESD) protection circuitry and output pads; a 16 Gb/s PRBS7 was transmitted with the driver being supplied with $V_{DD} = 0.9$ V. Voltage supplies ripples due to the series-terminated driver resulted to be higher (as shown in Figures 6.7a and 6.7b, by approximately 30 mV and 25 mV for V_{DD} and V_{SS}, respectively) than those caused by the shunt-terminated one, as expected [25].

The same test bench was then used to compare the eye diagrams from the two driver topologies, shown in Figure 6.7c, where the shunt-terminated driver is characterised by slightly slower rise and fall times, whereas eye height and width are equivalent for the two architectures.

A separate simulation was run to test the results from [129] concerning the frequency dependence of the driver's output impedance. Drivers with $R_{ON}/R_T \approx 0.1$ were designed as in [129] to drive a load of 300 fF to resemble realistic ESD protection circuitry and output pads, and their output impedance was simulated resulting in the plot of Figure 6.7d, where output impedance of shunt-terminated drivers features a slightly reduced bandwidth due to the additional contribution of the driver's MOS parasitic capacitances, as discussed in [129].

Power consumption and supply noise considerations led to choosing drivers with shunt terminations for the 16 Gb/s transmitter, also considering that other metrics (e.g. eye quality or bandwidth) did not result too degraded with respect to the series-terminated one.

The final step in the design of the driver was then to determine the ratio between the MOSS' on resistance and termination resistor R_{ON}/R_T. This was done by measuring both DC and AC R_{ON} variation for a number of MOSS of different size when V_{DS} was swept covering a range of $V_{DD}/4$, as well as by measuring variations in output voltage amplitude during regular operation. Figure 6.8a shows that both DC and AC R_{ON} become less dependent on V_{DS} as such a voltage decreases, or equivalently as R_{ON}/R_T is reduced. The choice of transistors size has an effect on the output voltage amplitude as well, as reported in Figure 6.8b, where the differential output swing is shown to shrink as the MOSS become smaller.

The sizing of the driver's transistor was completed by analysing the effect of PVT variations on the overall driver's output impedance (not shown). This was done after setting at 32 the number of driver slices for application of the FFE, and by making sure that even in the worst-case corners the total driver impedance was below 50 Ω in order for the impedance calibration circuit to always be able to restore the nominal 50 Ω impedance, since the chosen technique allows only to increase the output impedance (refer to Section 6.1.6 for further details on calibrating the driver's impedance). Finally, following considerations on the aforementioned aspects, the transistors were sized to provide $R_{ON} = 15\,\Omega$, whereas the termination resistors were set to $R_T = 20\,\Omega$.

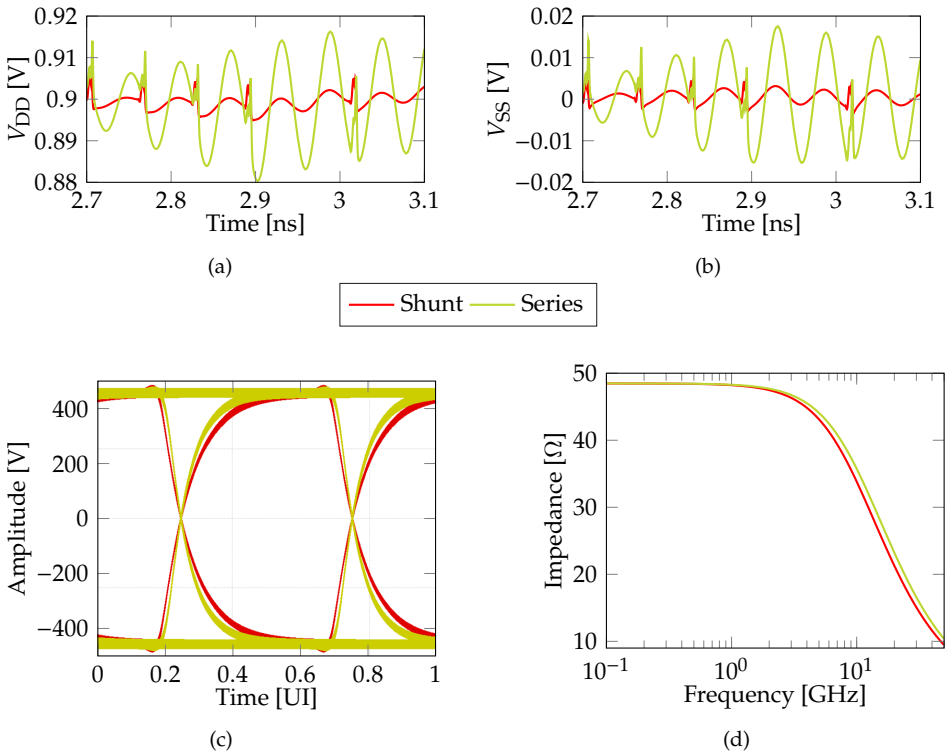


Figure 6.7: Simulation results comparing shunt-terminated and series-terminated VM p-over-n drivers: (a) and (b) noise on V_{DD} and V_{SS}, respectively, induced by the driver's switching activity; (c) eye diagrams; (d) overall termination impedance as a function of frequency. (a), (b) and (c) are simulated with PRBS7 at 16 Gb/s and the RLC supply network of Figure 6.6.

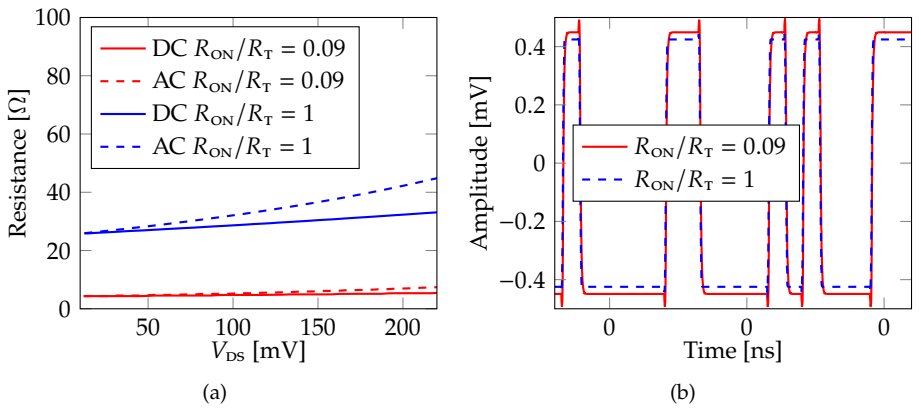


Figure 6.8: Example of simulation results to determine the driver's R_{ON}/R_T ratio: (a) DC and AC R_{ON} variation as a function of V_{DS}; (b) differential output voltage amplitude variations depending on R_{ON}/R_T. Two extreme cases are shown, with $R_{ON}/R_T = 1$ and 0.09.

6.1.4 Serializer

As briefly mentioned in Section 1.3.4, the serializer plays a major role in the design of a transmitter: It is directly linked to the working frequency of the digital macro (the first stage of the serializer has to operate at its same frequency), from which it receives the input parallel data, and has to comply with the timing constraints imposed by the data rate of the transmitted serial data stream, as well as provide clean full-rate data to the pre-driver and the subsequent driver. Implementation of a half-rate serializer benefits from the halved clock frequency due to the removal of the last retiming stage (see Figure 1.21), which makes the timing constraint of the critical path less stringent, although still challenging at such high data rates.

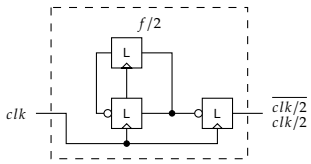
Designing a half-rate 40:1 serializer means implementing two parallel 20:1 serializers (one for the even and one for the odd path, see Figure 6.1), each providing 8 Gb/s data to the final multiplexing stage that transfers the full-rate data stream to the pre-driver/driver.

Operation of the 40:1 serializer requires clocks at different frequencies, from a low frequency one to process the parallel data coming from the digital macro to one at half the data rate for the final multiplexing stage before the driver. The 8 GHz clock is synthesised outside the SerDes as a differential signal and then divided as necessary inside the transmitter: For the chosen serializer architecture, clocks at four different frequencies (8, 4, 2 and 0.4 GHz) are required, so that the input clock has to be sequentially divided by 2, 2 and 5.

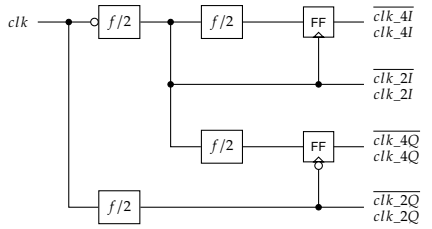
The first two such frequency divisions are performed using the differential latch-based frequency dividers of Figure 6.9a combined as in Figure 6.9b to naturally operate on differential signals, thus providing four quadrature clock phases. Additionally, the 2 GHz clock is resampled by the 4 GHz one so to align the edges of corresponding phases at different frequencies, hence ensuring that timing margins in the serializers are not degraded by frequency division. All clocks are then properly buffered so to ensure the required driving strength for the various loads while maintaining their relative phase relations.

The division by five is performed with a circuit resembling a linear feedback shift register (Figure 6.9d), as proposed in [130]; two such 5:1 dividers were instantiated, fed with $\overline{clk\_4I}$ and clk_4I, so to balance the load at their output (as well as the one experienced by the stages preceding them) and provide two sets of four clocks each ($abcd$ and $abcd'$) having a fixed time offset for the various 5:1 serializers (Figure 6.9e).

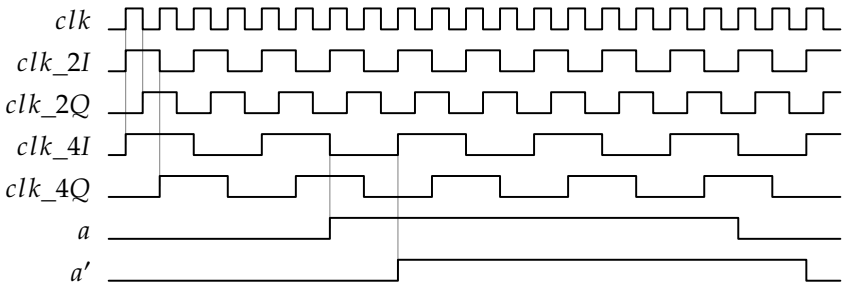
The 40:1 serializer implements regular (2 flip-flop, 1 latch) 2:1 serializers (Figure 6.10a) for the last two stages preceding the final MUX, whereas the first serialization level is composed of 5:1 serializers (Figure 6.9e) taken from [130] and is followed by retiming flip-flops to align the data for the following stages; half a serializer path is shown in Figure 6.11 and its timing diagram is presented in Figure 6.12. Selection of FFE taps (Figure 6.10b) is achieved simply by resampling the data twice (the first time for retiming purposes, the second time to generate the delayed sample, which provides the pre-cursor correction) and propagating the main- or pre-tap (depending on the user's input via FFE_sel) through a 32-times replicated multiplexer to drive each driver's segment. This is performed at quarter rate to reduce power consumption due to switching at the maximum available clock frequency, as would be the case if it were implemented immediately before the final multiplexing stage: More precisely, schematic simulations proved that the chosen location for the FFE tap selection yields approximately 50 % higher *average*



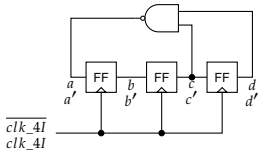
(a) By-2 frequency divider.



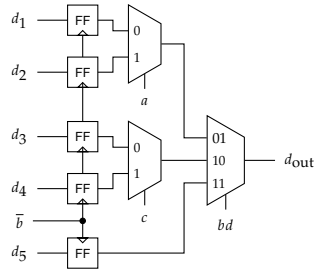
(b) Quadrature by-2 and -4 frequency divider.



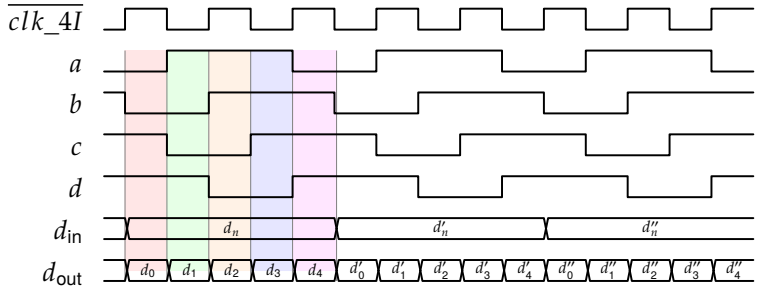
(c) Timing diagram of the clocks used in the 40:1 serializer; a and a' are generated with the circuit of plot (d) from $\overline{clk\_4I}$ and clk_4I, respectively.



(d) By-5 frequency divider [130] generating selection signals $abcd$ (and $abcd'$) for the circuit of Figure 6.9e.

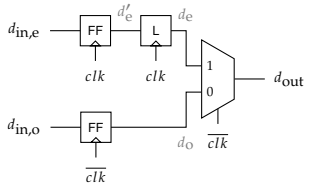


(e) 5:1 Serializer [130] with input resampling; selection signals a, b, c and d are provided by the circuit of Figure 6.9d.

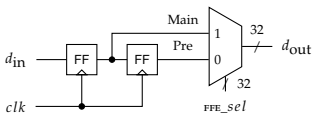


(f) Timing diagram of the 5:1 serializer; d_in represents the 5-bit input vector.

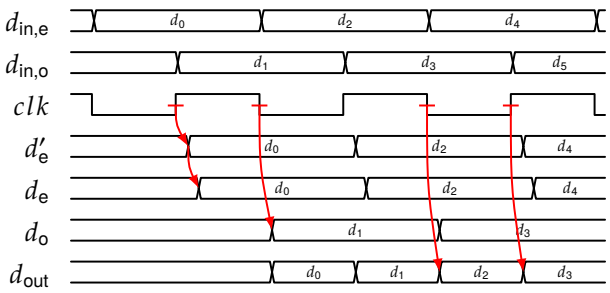
Figure 6.9: Frequency dividers for the 40:1 serializer, 5:1 serializer and corresponding timing diagrams. The circuits in Figures (a) and (b) are drawn as single ended for simplicity, although they are differential.



(a) 2:1 serializer stage.



(b) 2-tap FFE selection stage.



(c) Timing diagram of the 2:1 serializer stage.

Figure 6.10: Relevant building blocks for the 40:1 serializer with timing diagrams; all lines are differential except the d_{out} of the 2:1 serializer that feeds the final MUX in Figure 6.11.

power consumption (11.6 mW vs. 7.7 mW), but reduces power *peaks* by more than 40 % (25.4 mW vs. 43.7 mW), which was considered more relevant in terms of consequences for signal integrity and PSIJ. The last 2:1 serializer performs single-ended to differential conversion for the final MUX to work on differential signals.

Serializers typically employ a single clock phase (possibly with both polarities, depending on implementation of the digital circuits) for each serialization stage, so that sampling margins are constrained by the propagation delay of digital logics and by the edge alignment among the various clocks. In order to alleviate such timing constraints and to reduce latency as much as possible, the proposed 40:1 serializer requires four quadrature phases for both the 4 GHz and 2 GHz clock, whereas two phases separated by 250 ps (half period of the 2 GHz clock) are enough to generate the 400 MHz clocks driving the 5:1 serializers; refer to Figures 6.9c and 6.9f for the clocks' timing. By properly applying the various clock phases to different stages of the 40:1 serializer (refer to Figures 6.11 and 6.12 for the even 20:1 serializer), all data are guaranteed to be sampled in the middle of their own unit interval, hence satisfying both setup and hold times of the flip-flops.

Schematic simulations were run over automotive PVT corners to check the robustness of the serializer in terms of clock phases separation/quadrature, clock duty cycle and timing margins at the various sampling stages. The nominal values and worst-case results are reported in Table 6.1, showing the accuracy of the frequency division in terms of quadrature and separation between edges of corresponding 2 and 4 GHz clock phases, as well as by the resulting duty cycle; the sampling margins are divided by data rate and are shown to be large enough (so that the clocks' edges are sufficiently far from transitions) by ensuring that the data to be sampled toggles as much as possible at mid-period of the sampling clock. Such safe margins hold particularly for half- and quarter-rate data (ide-

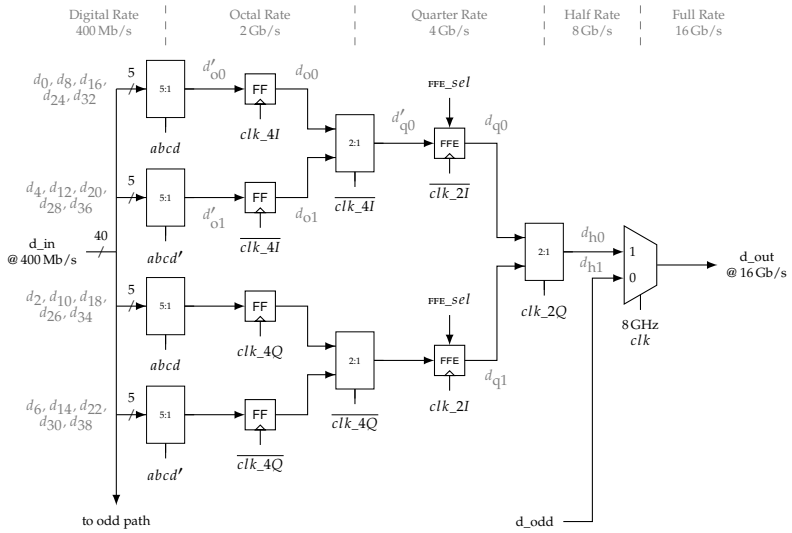


Figure 6.11: Simplified schematic of the even path of the 40:1 serializer designed for the half-rate transmitter; the odd path has the same structure; all lines are single ended until the half-rate section, where d_odd, d_even, d_out and the 8 GHz clk are differential signals. clk_2I/Q and clk_4I/Q are generated from the frequency divider shown in Figure 6.9, $abcd$ and $abcd'$ are generated by the circuit of Figure 6.9d from $\overline{clk\_4I}$ and clk_4I, respectively, whereas the 8 GHz clk is generated outside the HSIO. The 5:1 and 2:1 serializers, and the FFE switch matrix are reported respectively in Figures 6.9e, 6.10a and 6.10b. The timing diagram of the serializer is shown in Figure 6.12 using the signals annotated in gray.

ally, they should be sampled at 62.5 and 125 ps from the input data transitions, respectively), whereas octal-rate data are somewhat skewed from their ideal timing (250 ps); nonetheless, setup margin is the same as for quarter-rate data (the minimum values are slightly higher than 90 ps in both cases).

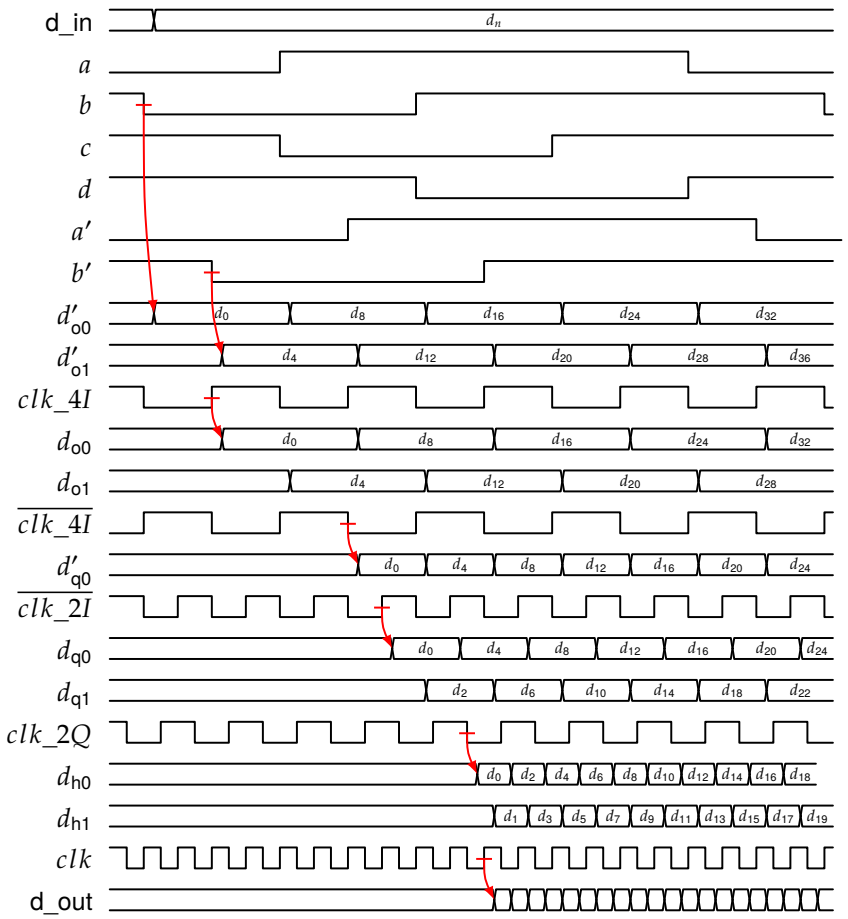


Figure 6.12: Timing diagram of the 40:1 serializer designed for the half-rate transmitter; the signal considered belong to the even half serializer of Figure 6.11.

Measure	Unit	Nominal	Min	Max
Clock quadrature: 2 GHz	ps	125	123.7	124.4
Clock quadrature: 4 GHz	ps	62.5	63.46	64.19
Clock separation: 2 and 4 GHz	ps	0	−1.9	−0.2
Clock duty cycle: 2 GHz	%	50	49.84	50.88
Clock duty cycle: 4 GHz	%	50	49.75	50.07
Margin[a]: Octal-rate data	ps	250	90.81	294.1
Margin[a]: Quarter-rate data	ps	125	90.96	140.1
Margin[a]: Half-rate data	ps	62.5	58.19	64.46

[a]Time difference between transitions of the input signal and rising edges of the sampling clock.

Table 6.1: Relevant timings and timing margins of the 40:1 serializer from schematic simulations over PVT variations: SS, SF, FS, FF process corners; V_{DD} = 0.81 V, 0.9 V and 0.99 V; temperatures of −40 °C, 25 °C and 165 °C.

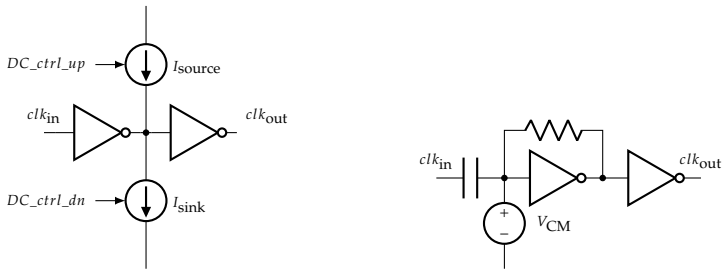
6.1.5 Duty Cycle Correction Circuit

When implementing half-rate transmitters, one of the most critical blocks is the final 2:1 MUX [25, 36, 39, 93]: The clock that is used to toggle between even and odd data must be properly centred so that output jitter depends not on the timing and slope of the data transitions, but only on the phase noise of the transmitter's half-rate clock; moreover, its duty cycle must be 50 % to avoid injecting DCD into the transmitted stream of data, which cannot be tracked by the CDR and reduces eye width at the transmitter.

This mandates correction circuits to restore 50 % duty cycle in the half-rate clock so to reduce the aforementioned issues related to DCD. Such techniques can be approximately split into two categories based on the correction method employed: *Edge* or *common-mode* regulation, the working principles of which are schematically shown in Figure 6.13. Both techniques are implemented in inverter chains devoted to clock distribution; however, while edge regulation is based on injecting current to/drawing current from a node in order to improve either the clock's rise or fall time, hence modifying the duration of the clock's high and low levels [39,93], common-mode regulation modifies an inverter's trip point, thus anticipating or delaying the moment when a transition of the input is detected and propagated [36].

Since the HSSI literature is quite scarce of in-depth analysis or comparisons of the aforementioned two techniques, a few simulations with the "ideal" circuits of Figure 6.13 were run to gain a first-order estimate of their different behaviour. As shown in Figure 6.14, common-mode regulation resulted in less linear control of the duty cycle correction, as well as more degradation of the output clock signal, in terms of jitter and shape of the waveform; moreover, typical common-mode regulation schemes tend to be slightly more involved from a design standpoint (e.g. see [36]), while edge-regulation architectures require less effort and provide straightforward control of the output clock's duty cycle.

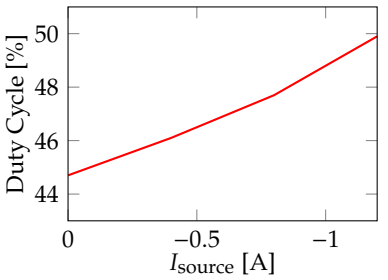
An edge regulation-based duty cycle control circuit was hence implemented based on [39,93] and is shown in Figure 6.15: Current-starved inverters are placed in parallel to the regular inverters of a clock buffering chain so that they can modify the clock's edges and thus affect its duty cycle; the current through the starved inverters is controlled by enabling up to 16 thermometer-encoded transistors connected to the voltage supplies.



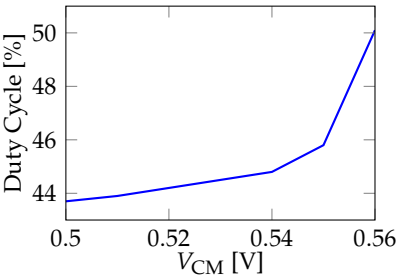
(a) Edge regulation

(b) Common-mode regulation.

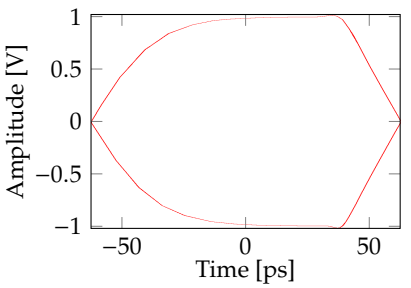
Figure 6.13: The two most common approaches to duty cycle correction. The simplified schematics are shown as single ended, although they are typically implemented as differential.



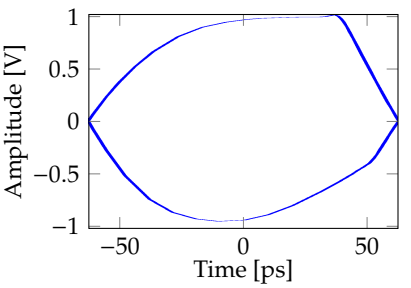
(a) Edge regulation: Control current vs. clock's duty cycle.



(b) Common mode regulation: Control voltage vs. clock's duty cycle.



(c) Edge regulation: Output eye diagram at 50 % duty cycle.



(d) Common mode regulation: Output eye diagram at 50 % duty cycle.

Figure 6.14: Simulations of the two most common duty cycle correction architectures (refer to Figure 6.13 for their schematics) run with an 8 GHz clock: Duty cycle characteristics vs. control quantity for (a) the edge-regulation architecture and (b) the common mode-regulation one; resulting eye diagrams when the clock's duty cycle is restored to 50 % for regulation with (c) edges and (d) common mode.

The design achieved a correction range of approximately ±6 % by implementing 16 current-starving transistors to control the duty cycle, translating to a nominal resolution of less than 500 fs for an 8 GHz half-rate clock.

Duty cycle detection is rather straightforward and is based on the architecture shown in [93], involving a simple RC low-pass filter that feeds a voltage comparator with a filtered version of the differential clock resembling a DC voltage, so that deviations from differential 0 V are an indication of the duty cycle affecting the half-rate clock. The comparator is clocked with a low-frequency clock (the one divided by 40 that is employed in the digital circuits [55]) and its output is sent to the digital macro where synthesised logics[5] then controls the current-starved inverters to restore 50 % duty cycle.

Given the half-rate clock frequency, the low-pass filter was designed for a cut-off frequency of 8 MHz, so to provide nominally −60 dB gain at 8 GHz. This translates to a differential amplitude of the filtered clock of approximately 2 mV, hence to a duty cycle detection resolution slightly larger than 0.2 %, with $V_{\text{DD}} = 0.9\,\text{V}$. In order to profit of such a resolution, a differential comparator with very low input

[5]Not implemented yet at the time of writing.

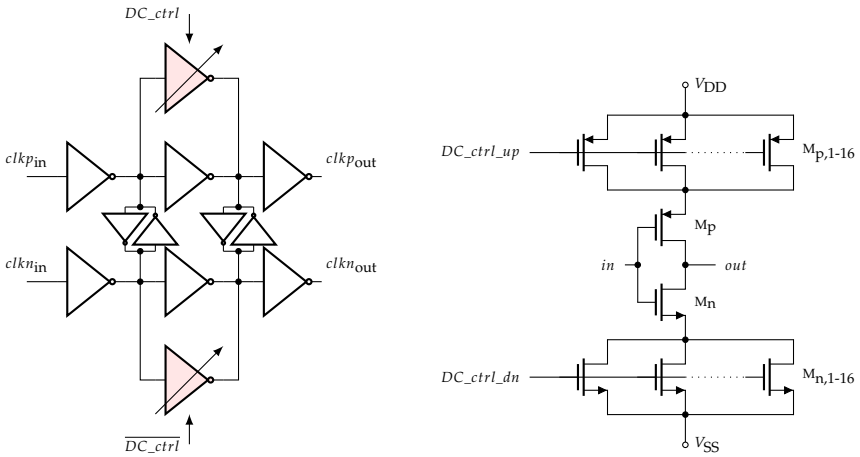


Figure 6.15: The implemented edge regulation-based duty cycle correction circuit, showing (a) the overall clock buffering inverter chain and (b) the current-starved inverters that provide edge regulation (shaded in the left schematic).

offset voltage is mandatory: A StrongARM topology [131] was chosen and designed for $V_{os} \approx 1.5\,\text{mV}$ by reducing as much as possible the offset due to the input differential pair, while making sure that the cross-coupled pairs contribute negligibly to the input offset voltage; having achieved such a specification, as verified by schematic simulations, no offset calibration technique was implemented.

6.1.6 Driver's Impedance Calibration

Setting the driver's output impedance to match the characteristic impedance of the channel is critical to improve return loss and reduce reflections [39], as briefly mentioned in Section 1.3.2. The most straightforward method to keep the transmitter's output impedance under control is to *segment* the driver into *slices* connected in parallel, then enabling as many slices as are required to meet the target output impedance specification [39,108,132]. However, variations typical of IC technology entail that the amount of slices instantiated needs to be large enough to tune the impedance on all corners and provide a good calibration resolution, thus requiring more power-hungry pre-drivers and clock buffers due to the high gate capacitance of the parallel driver segments; moreover, impedance tuning with slices is typically a foreground technique and also cannot compensate mismatches in NMOS and PMOS impedances [39,108].

A more advanced approach involves the analog loop depicted in Figure 6.16, which can operate in the background by exploiting a replica of the driver to tune the MOS' impedance: The two regulation loops tune the controlled resistors $M_{reg,PU}$ and $M_{reg,PD}$ so that the impedances seen looking towards the pull-up and pull-down nodes, respectively Z_{PU} and Z_{PD}, are both $50\,\Omega$ and, consequently, $V_{DD}/2$ is dropped across the $100\,\Omega$ replica channel load [39, 108]. The most critical aspects of this technique are related to the fact that the resistance provided by the regulation MOSS has to cover a range large enough to compensate for variations in the driver's

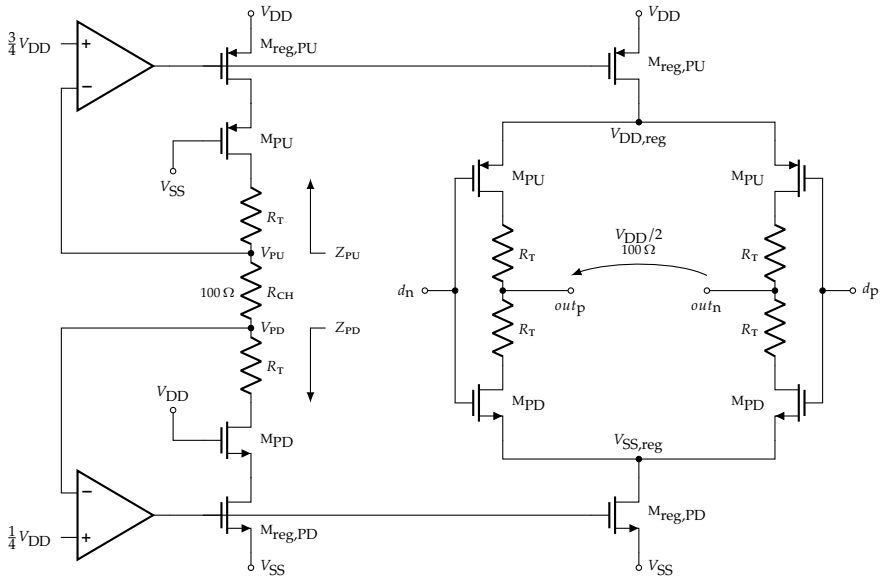


Figure 6.16: Analog-loop based driver's impedance calibration scheme, showing the replica slice with the regulation loop on the left and the actual differential driver on the right.

transistors and termination resistances [39, 108]: If a wide range of output swing has to be supported, it is difficult to ensure that the driver's MOSS operate in the linear region due to short-channel effects and reduced supply voltage of advanced technology nodes; the MOS' V_{DS} changes with the on resistance variation because the drop across the channel impedance is fixed by the regulation loop, and the MOS' target AC and DC resistance deviation is higher as V_{DS} increases, making simultaneous swing and return loss control not achievable; moreover, only DC driver impedance can be controlled by typical analog impedance calibration loops.

Impedance calibration based on analog control loops was chosen to enable tuning operations in the background (thus being able to compensate also supply voltage and temperature variations [108]) and to reduce area consumption and large gate capacitance typical of segmented drivers. Conventional single-ended, two-stage compensated operational amplifiers (*OpAmps*) were designed for the feedback loops: The one employed in the pull-up loop is shown in Figure 6.17 and employs NMOSS in the input differential pair, whereas the other one has complementary topology as it is based on PMOSS. The design of such OpAmps was mainly targeting low input offset voltage V_{os} to accurately track the loop's reference level and enough DC gain to set the driver impedance so that the whole gate voltage range (from 0 V to $V_{DD} - V_{th}$ for $M_{reg,PU}$, from V_{th} to V_{DD} for $M_{reg,PD}$) can be covered with a difference of less than 10 mV in the OpAmps' input, i.e. between any of the two reference voltages and the corresponding pull-up/down voltage. The former specification was set by considering that, e.g. for the pull-up,

$$V_{PU} = V_{DD} \frac{150\,\Omega}{150\,\Omega + Z_{PU}} \tag{6.1}$$

holds, where 150 Ω is the combined resistance of the pull-down slice (assumed to

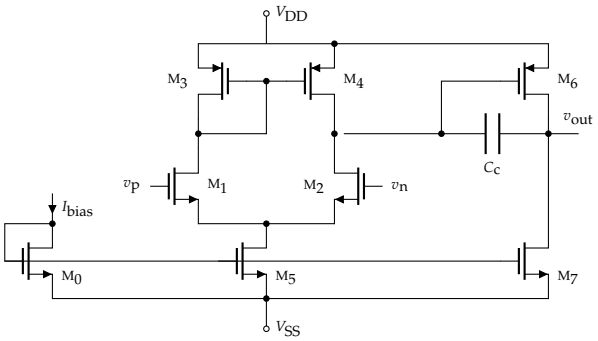


Figure 6.17: Operational amplifier based on nMOS differential pair for the pull-up loop in the impedance calibration circuit. C_c is the compensation capacitor, I_{bias} is the bias current for the current mirror generated from another circuit (not shown).

be perfectly matched at 50 Ω) and channel (assumed to be perfectly terminated), while Z_{PU} is the regulated impedance of the pull-up (which targets 50 Ω). From Figure 6.16, the pull-up voltage can be expressed as the sum of the reference $\frac{3}{4}V_{DD}$ and the OpAmp's offset voltage V_{os}, so that from Equation 6.1 the pull-up impedance can then be expressed as

$$Z_{PU} = 150\,\Omega\left(\frac{V_{DD}}{V_{PU}} - 1\right) = 150\,\Omega\left(\frac{V_{DD}}{\frac{3}{4}V_{DD} + V_{os}} - 1\right) \in [43.0\ 57.5]\ \Omega, \tag{6.2}$$

where a V_{os} spanning $\pm 3.5\sigma_{os} = \pm 3.5 \cdot 7\,\text{mV}$ was assumed to compute the impedance range; the extent of such a variation for the OpAmps' input offset was deemed to be acceptable in terms of the consequent driver's impedance variation and was thus used as a design specification. The latter specification entails that a gain of approximately 38 dB would suffice to cover the whole input range of the regulation transistors with less than 10 mV deviation from the reference voltage, given the nominal $V_{DD} = 0.9\,\text{V}$ and an estimate of the MOS' threshold voltages obtained through simulations:

$$G_{spec} = 20\log\left(\frac{V_{DD} - V_{th}}{10\,\text{mV}}\right) = 20\log\left(\frac{900 - 100}{10}\right) \approx 38\,\text{dB}. \tag{6.3}$$

Post-layout Monte Carlo simulations result in an input random offset of about 6.488 mV for the p-based OpAmp and 6.733 mV for the n-based one, hence complying with the specifications that were set, while input systematic offset is 747 µV and −641 µV, respectively. The worst-case DC gains for the two OpAmp versions are, respectively, 36.7 dB and 44 dB; although the p-based OpAmp gain is slightly below the 38 dB specification in some PVT corners, simulations of the whole impedance calibration circuit proved it to be high enough (shown later in this Section).

Design of the actual impedance calibration loop was performed by considering a reference channel impedance $R_{CH} = 3.2\,\text{k}\Omega$ (instead of 100 Ω shown in Figure 6.16) in order to take into account the multiplicity due to FFE segmentation of the driver; the two reference voltages for the analog loop were implemented through a resistor ladder to equally divide the voltage supply with the desired granularity. The regulation transistors $M_{reg,PU/PD}$ operate on a per-slice basis (hence requiring 32

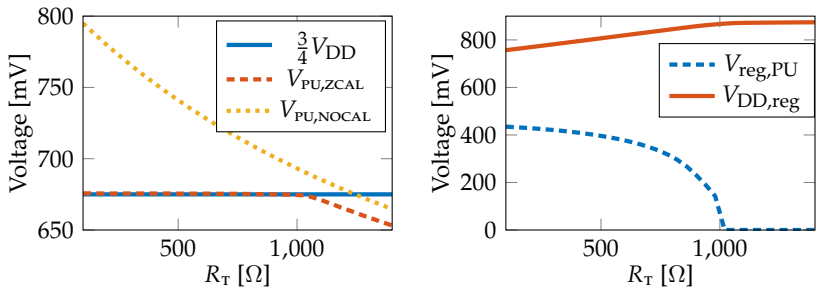


Figure 6.18: Operating-point simulations of the impedance calibration loop with post-layout OpAmps for the pull-up loop: The supply voltages V_{DD} and V_{SS} are respectively 0.9 V and 0 V, R_T is varied from 100 Ω to 1400 Ω and the pull-down is also being calibrated, albeit not shown. $V_{PU,ZCAL}$ is the voltage of the calibrated pull-up output, $V_{PU,NOCAL}$ is the voltage of an uncalibrated one and $V_{reg,PU}$ is the gate voltage of the regulation pMOS, $M_{reg,PU}$. The calibration loop ensures $Z_{PU/PD} = 50\,\Omega$ (i.e. $V_{swing}/V_{DD} \approx 1$) as long as R_T does not exceed approximately 1150 Ω.

of them) on both the *p* and *n* halves of the driver in order to keep them always active, even when switching from one half to the other, and were sized so not to limit the current drawn by the driver when fully active.

Figure 6.18 shows operating-point simulations of the impedance calibration loop with post-layout OpAmps for the driver's pull-up, while the pull-down is also being calibrated (not shown, although results are analogous). The supply voltages are $V_{DD} = 0.9\,V$ and $V_{SS} = 0\,V$, the termination resistance R_T is swept from 100 Ω to 1400 Ω.

Figure 6.18a shows that the pull-up voltage $V_{PU,ZCAL}$ of the calibrated driver is kept close to the reference voltage $\frac{3}{4}V_{DD}$ unto approximately 1150 Ω (where a difference of more than 1 % between the two voltages appears), whereas an uncalibrated driver instantiated as a term of comparison shows a pull-up voltage $V_{PU,NOCAL}$ that directly depends on R_T. This is in line with expectations from the design of the driver (see Section 6.1.3): As the pull-up/down MOSs are designed to provide 480 Ω on resistance, and the ideal $Z_{PU/PD}$ for a single driver slice is $32 \cdot 50\,\Omega = 1600\,\Omega$, the impedance calibration loop is expected to stop working above 1120 Ω because, since it operates by making $M_{reg,PU}$ less conductive, it can only increase the driver's impedance, not reduce it.

The gate voltage $V_{reg,PU}$ of the regulation transistor $M_{reg,PU}$ is then shown in Figure 6.18b, highlighting the fact that supply regulation is possible until there is driving range to modulate such a pMOS resistance, i.e. until it is fully on. The positive supply $V_{DD,reg}$ seen by the driver's pull-up is also reported, clearly showing that when R_T is low, and thus the driver's voltage swing would be high, reducing the supply increases the driver's impedance and restores impedance matching with the channel; instead, if is too high, a reduced headroom prevents making the regulation pMOS $M_{reg,PU}$ more conductive, thus posing an upper limit to the termination impedance that can be calibrated [39, 108].

The effect of the impedance calibration loop on the driver's dynamic performance was investigated by running time-domain simulations with a PRBS31 input

$R_T[\Omega]$	t_{rise}[ps]	t_{fall}[ps]	$V_{DD,reg}$[mV]	V_{swing}/V_{DD}	$Z_{PU}[\Omega]$
100	15.2	15.3	754	1.012	49.0
300	13.9	13.9	780	1.012	49.0
500	12.8	12.7	806	1.010	49.0
700	11.7	11.7	832	1.010	49.2
900	11.2	11.1	857	1.006	49.6
1100	12.9	12.9	872	0.962	55.2
1300	19.9	20.0	873	0.908	61.4

Table 6.2: Time-domain simulations of the impedance calibration loop featuring post-layout OpAmps. A PRBS31 with rise/fall times equal to 10 ps is used as input to the driver, whose output is connected to a 100 Ω channel with 100 fF pad capacitance; the supply voltages are $V_{DD} = 0.9$ V and $V_{SS} = 0$ V; R_T is varied from 100 Ω to 1400 Ω; rise/fall times are averaged over ten transitions (maximum standard deviation: 0.3 ps). The calibration loop ensures $Z_{PU/PD} = 50\,\Omega$ (i.e. $V_{swing}/V_{DD} \approx 1$) as long as R_T is not too high (approximately 1150 Ω, as shown in Figure 6.18); shaded cells highlight results where R_T is close to or exceeds the aforementioned value and the impedance calibration fails.

having rise/fall times of 10 ps and a driver on a 100 Ω channel with 100 fF load capacitance to emulate output pads. The driver's rise and fall times were measured as the time span required to cover 90 % (5 % to 95 %) of the nominal swing V_{DD}, they were then averaged over ten signal transitions and reported in Table 6.2 as a function of R_T, along with the corresponding regulated supply $V_{DD,reg}$ and output V_{swing}/V_{DD} as an indicator of impedance matching.

The results show that rise and fall times approach optimal values the closer R_T is to the calibration loop's working limits (where the regulation transistors are almost fully on), whereas timing performance degrades the more $M_{reg,PU}$ is made less conductive in order to increase the effective termination impedance. As soon as the calibration fails to set $Z_{PU/PD} = 50\,\Omega$, rise and fall times increase because the driver's output RC time constant becomes the limiting factor for switching speed. The driver's pull-up impedance Z_{PU} was also extracted and shows good calibration within the tuning range covered by the circuit, with deviations from the ideal 50 Ω target likely due to the OpAmps' non-idealities.

6.2 Putting it all Together

The final step in the design of the transmitter is to verify that all the blocks described in the previous Sections can operate correctly when put together, and that the overall circuit can effectively respect the given system specifications. Unless otherwise specified, all simulations are run at 16 Gb/s with PRBS31.

One major impairment affecting high-speed circuits is PSIJ (see Appendix A.4 for further details), whose effects can be estimated by running simulations with a circuit mimicking an actual *power delivery network (PDN)* connecting the ideal $V_{DD,ideal}$ and/or $V_{SS,ideal}$ to the device under test. The RLC network featuring $R = 100\,m\Omega$, $L = 1$ nH and $C = 60$ pF of Figure 6.19 was instantiated to reproduce the effect of real power supply interconnections on the circuit. As shown in

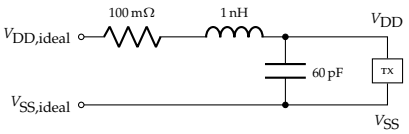
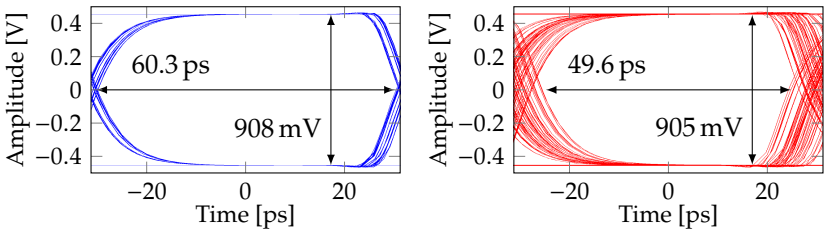


Figure 6.19: The PSN connected to the transmitter under test for PSIJ simulations.



(a) No PSN.

(b) With PSN.

Figure 6.20: Simulation results comparing eye diagrams (a) without and (b) with a power supply network featuring $R = 100\,\text{m}\Omega$, $L = 1\,\text{nH}$ and $C = 60\,\text{pF}$.

Figure 6.20 for the transmitter's output, the PSIJ increases the output peak-to-peak jitter from 2.2 ps to 13.1 ps; instead, the effect of power supply noise on the signal amplitude is rather low, as demonstrated by the fact that the eye height is reduced by merely 3 mV.

The effectiveness of the duty cycle correction circuit was checked by imposing a clock with 46 % duty cycle, manually[6] correcting it as close as possible to 50 % and verifying the quality of the resulting signal and transmitter's output. Figures 6.21a and 6.21b respectively show the distorted and the corrected 8 GHz clock after its duty cycle was brought to 49.8 %; Figure 6.21c shows the duty cycle-distorted driver's output eye diagram, whereas Figure 6.21d reports the same signal after the correction, demonstrating that the eye width was improved by 3.5 ps by removing the duty cycle distortion.

By comparing the eye diagrams of Figures 6.20a and 6.21d one may notice that the one corresponding to the 49.8 % duty cycle clock has a slightly larger width than the one obtained with an ideal 50 % duty cycle clock. This is an indication that performance is not limited by the clock's duty cycle as long as it is close enough to 50 %; instead, eye width is limited by other non-ideal effects (e.g. quality of the single ended to differential conversion and strength of the pre-driving stage) that might be investigated to further improve the transmitter's output quality.

The impedance calibration circuit was eventually tested in the full transmitter with post-layout OpAmps over automotive PVT variations to check possible large deviations from the expected behaviour. The resulting histogram is reported in Figure 6.22, showing that the great majority of runs are located in the vicinity of the ideal $V_{swing}/V_{DD} = 1$, whereas three runs (all at 175 °C) are situated below 0.97 down to approximately 0.88, corresponding to a worst case total driver impedance $Z_{PU} + Z_{PD} = 113.2\,\Omega$, i.e. less than 15 % away from the targeted $100\,\Omega$, although still capable of reducing reflections [46]. Despite such worst cases show that the

[6] As mentioned in Section 6.1.5, the digital correction algorithm has not been implemented yet.

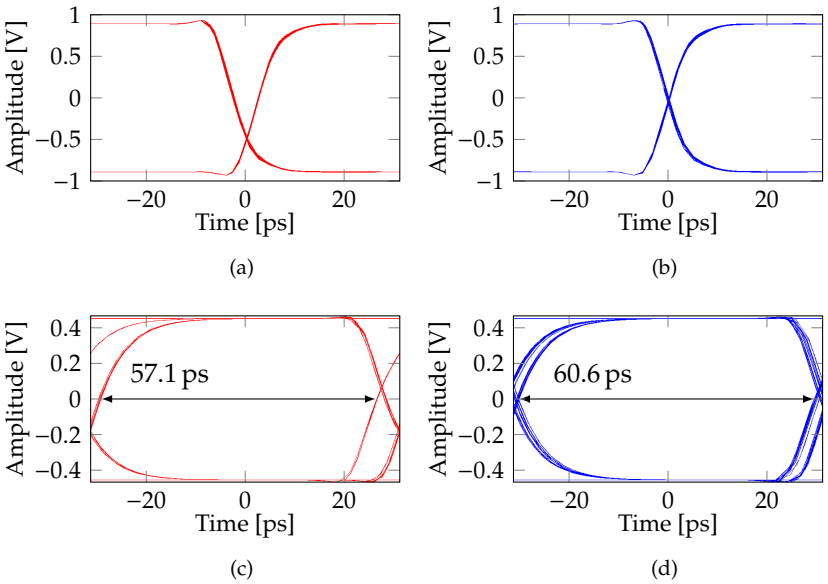


Figure 6.21: Simulation results of duty-cycle distortion and correction: Eye diagrams of 8 GHz clocks (a) with 46 % duty cycle and (b) after correction brought the duty cycle to 49.8 %; eye diagrams of the driver's output (c) with duty cycle distortion and (d) after correction.

calibration circuit is ineffective in these extreme conditions, the remaining 33 out of 36 runs are all in the range 100.0 Ω to 101.7 Ω, i.e. within 2 % from the target driver output impedance. This means that reducing the driver's sensitivity to PVT variations is key to the correct functioning of such a calibration technique over the whole automotive range.

Concluding, Table 6.3 compares the transmitter designed in this Chapter with

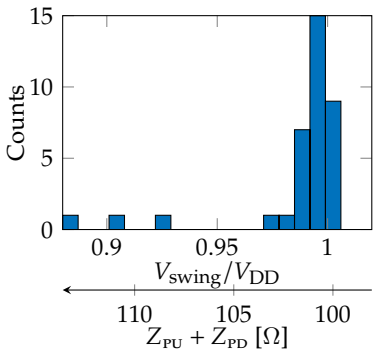


Figure 6.22: Histogram of driver's normalised output swing (V_{swing}/V_{DD}) and output impedance ($Z_{PU} + Z_{PD}$) over PVT variations with post-layout OpAmps: SS, SF, FS, FF process corners; V_{DD} = 0.81 V, 0.9 V and 0.99 V; temperatures of −40 °C, 25 °C and 175 °C.

other works targeting similar data rates; notice that performance in terms of power consumption for this work could be computed only from schematic simulations without post-layout parasitics, although parasitic gate capacitances estimated from the technology specifications where included.

	This work	[133]	[79]	[108]	[134]
Node [nm]	28	40	28	65	28
Speed [Gb/s]	16	14	22	20	18
Signalling scheme	PAM-2	PAM-2	PAM-2	PAM-2	PAM-2
Driver	VM	VM	CM	VM	CM
Impedance calibration	Analog	Segments	None	Segments	None
V_{swing} [mV]	900	n.a.	500	250	n.a.
FFE taps	3	4	2	2	3
Power [mW/(Gb/s)]	1.12[a]	7.25[b]	5.9[c]	1.53[d]	5.38[d]
V_{DD} [V]	0.9	1.1	0.9, 1.35	1.2	1.2

[a] TX, schematic only
[b] TX +RX+PLL/2
[c] TX +RX
[d] TX +PLL

Table 6.3: Transmitter performance at 16 Gb/s and comparison with similar works.

Chapter 7

Conclusions and Future Work

This thesis presented research work in the field of high-speed wireline transceivers for chip-to-chip communication in the automotive environment, focussing on systems for fully-adaptive equalization that automatically configure the equalizers to optimally compensate the inter-symbol interference introduced by the channel, even if its characteristics are not known. This was done starting with system-level models and analyses, moving to digital and analog/mixed-signal design, eventually leading to experimental characterisation of the 12 Gb/s test chip fabricated in 28 nm planar CMOS technology.

At first, a probabilistic model was developed in Octave/MATLAB to efficiently simulate HSSIS while taking into account the ISI induced by the transmissive medium as well as the behaviour of various analog blocks that compose the transceiver, including the equalizers. This allows an assessment of the SerDes performance based on observation of various figures of merit, such as the eye diagram and the bathtub plot at the receiver's slicers, or simply the channel's pulse response.

Such a modelling approach was then employed to include the adaptation of equalizers through sign-sign least-mean squares algorithms, after developing the required theoretical framework to understand how this type of adaptive loop applies to FFE, DFE, sampling phase and to CTLE (when interpreted as an FIR filter). Considerations on the practical implementation of such systems as digital circuits were proposed and simulation results for a variety of equalizers and configurations were presented. As a conclusion of this modelling activity, a behavioural Verilog-A version of the SS-LMS algorithm was instantiated in a circuit simulator to test adaptation on the transistor-level design of a real wireline transceiver (comprising more second-order and non-linear effects than what can be efficiently implemented in the probabilistic model) and assess the prediction capabilities of the Octave/MATLAB model with respect to the circuit simulator when dealing with high-loss channels typically considered in modern automotive standards.

Then, before starting the development of the new chip with fully-adaptive equalization, the old one was verified in simulation and characterised in the laboratory over automotive corners to evaluate performance and possible critical points, so to determine which blocks required a re-design or simply an upgrade. Subsequently, a few circuits (among which the FFE switch matrix and parts of the clock distribution network) were modified to improve performance while still targeting a 12 Gb/s data rate with PAM-2 in 28 nm planar CMOS technology, whereas other upgrades were required simply for the fully-adaptive algorithm to operate (chiefly, addition of a variable-threshold comparator and error samples deserialization). Moreover, a digital eye monitor was developed to visualise the analog signal at the input of the slicers regardless of the half-rate, 1-tap speculative DFE architecture.

The fully-adaptive algorithm was implemented in VHDL to operate on the DFE taps and on the slicers' sampling phase, and extensively verified on a behavioural version of the HSIO that was used for characterisation, so to test the various possible settings and configurations of the SS-LMS loop. Results obtained with such an RTL block were compared with those from the probabilistic model (the Octave/MATLAB simulator to obtain the eye diagram with ISI) and from the circuit simulator, verifying the overall correspondence of the three methods, despite small discrepancies due to obvious differences in the considered approaches.

The fabrication and the following characterisation of the transceiver marked the conclusion of the work on fully-adaptive equalization and provided the opportunity to evaluate the modelling and design activities performed up to that point. The HSIO was tested on different channels featuring various loss and reflection coefficients by running the SS-LMS algorithm and acquiring bathtub plots down to BER = 10^{-12} to verify that the configuration found by the adaptive loop actually resulted in optimal opening (i.e. other settings could not provide better performance); additionally, the SS-LMS algorithm was run with various settings to investigate the dynamics of adaptation.

Having completed the work on fully-adaptive equalization, the final part of the PhD was devoted to future developments aimed at improving the performance of high-speed wireline transceivers. The transmitter was regarded as a major bottleneck for achieving higher data rates, so that it was redesigned from scratch in order to target PAM-2 transmission at 16 Gb/s in 28 nm planar CMOS technology: A half-rate architecture was chosen to reduce power consumption and issues related to high-frequency clocks and, after having investigated and discarded the possibility to move to a DAC-based transmitter (due to concerns about power consumption), the driver was implemented as a voltage-mode p-over-n topology, a duty cycle correction circuit was added in the clock distribution network and driver's impedance calibration was implemented with an analog tuning loop; the serializer was designed with a custom approach to exploit multiple clock phases in order to reduce latency and relax timing constraints.

Such an increased data rate will require to reconsider the receiver, especially regarding its analog front end (currently made up of a VGA, a CTLE and the DFE's summing stage) and its clock distribution network, since the jitter budget shrinks at higher speeds. Moreover, providing a second-order CDR in place of the current first-order one will enhance the receiver's frequency-tracking capabilities.

Complementary activities focussed on another relevant aspect in high-speed wireline transceivers: Jitter, particularly that arising in the clock- and data-recovery unit. Simple analytical models were developed with in order to gain insight in the phenomenon and thus guide system-level design, rather than to provide accurate predictions of second-order and non-linear effects. Nonetheless, such an approach proved effective when compared with the results of a custom event-driven simulator developed in order to provide a versatile tool to test various CDR architectures.

Extension of the activity on the CDR with a focus on second-order loops and ISI-induced (hence data-dependent) jitter has already started and will provide further understanding on such a relevant topic in modern SerDes, whereas inclusion of power supply-induced jitter in the behavioural models of the overall HSSI has begun and has the potential to become an important tool in the design of advanced transceivers. Moreover, much of the work on modelling can be extended to comprise PAM-N signalling and initially lead to the design of a PAM-4 transceiver.

Appendix A

Modelling of Jitter in High-Speed Wireline Transceivers

As was briefly mentioned in Section 1.3.1, jitter (the uncertainty on the edge placement of a waveform) is a parameter of paramount importance when modelling and designing high-speed wireline transceivers, for it can affect the overall performance and increase BER at the receiver: In the case of a jittered clock, sampling is performed no longer where set by an ideal CDR (e.g. the point for which $h_{-0.5} = h_{0.5}$ holds in the case of an Alexander phase detector), so that the incoming signal may be sliced close to the transitions where there is less margin for a correct decision; alternatively, a jittered signal is said to produce a narrower eye diagram that makes a correct sampling difficult.

Jitter on the received signal can be due to two different causes: Jitter in the clock that generates the data stream at the transmitter and variations in the transitions' slope due to ISI. The former can be analysed by considering the quality of the transmitter's clock source and distribution network, whereas the latter is due to the features of the channel and depends on the transmitted sequence. At the receiver, jitter affects the HSSI depending on the quality of the receiver's clock distribution network and on the jitter-tracking capability of the CDR, which determines how much of the jitter in the received signal will be compensated when reconstructing the clock. The remaining, untracked jitter, plus other contributions introduced by the CDR itself, will affect sampling and possibly increase the BER. Therefore, designers must understand the various causes of the phenomenon and the ways it can degrade performance, but the capability of modelling and predicting jitter is required to gain a quantitative estimate of its effect and account for it in all design phases of wireline transceivers.

A basic theoretical framework for jitter is outlined in Appendix A.1, mainly to provide a mathematical foundation and define some relevant quantities, whereas the subsequent Appendix A.2 proposes a model to determine jitter in CDRs based on *bang-bang* phase detectors when the received signal used to determine the data sampling phase is not affected by ISI; the following Appendix A.3 deals with the case with ISI introduced by the channel and presents a simulator developed for such a purpose. Although not related to CDRs, Appendix A.4 deals with power supply-induced jitter that affects clocks and signals through the noise present in the voltage supplies and mostly due to the switching activity of the same circuits that suffer from it.

A.1 Jitter in Wireline Transceivers

A.1.1 Basics on Clock, Phase Noise and Jitter

A generic clock signal is usually expressed for modelling purpose as a purely sinusoidal waveform [44, 135]:

$$v(t) = [V_0 + \varepsilon(t)]sin\left(\omega_0 t + \phi(t)\right), \tag{A.1}$$

where V_0 is the nominal peak amplitude, $\varepsilon(t)$ is the amplitude error, ω_0 is the nominal angular frequency and $\phi(t)$ is the *phase error*; common assumptions for precision clocking systems are that $|\varepsilon(t)| \ll V_0 \; \forall t$ and that $\omega_0 t + \phi(t)$ is monotonically increasing with t.

Of major importance are the zero-crossing times t_n (i.e. those corresponding to $v(t) = 0$), for which the phase error expression

$$\phi(t_n) = n\pi - \omega_0 t_n \tag{A.2}$$

holds for each integer n. Since typical circuits driven by the clock change state only at the zero-crossing times, and due to other considerations, usually $\phi(t)$ represents the only significant non-ideal feature of the clock [135].

Despite the sinusoidal clock shown in the above treatment, it is common practice to use squared-up sinusoids as clock waveforms in order for them to switch as abruptly as possible [135]. However, the sinusoidal and squared-up signal models show nearly equivalent spectra for frequencies from 0 up to well beyond ω_0, and since most circuits are sensitive to clock crossings only, the sinusoidal approach can be employed to provide simpler descriptions of the phenomena involved [135].

Usually, in any application either the positive-going or the negative-going zero-crossings are used to mark a set of times $\tau_n \triangleq t_{2n} = nT_0$ when circuits can change state, where $T_0 = 2\pi/\omega_0$ is the nominal period, so that

$$\tau_n = nT_0 + \Delta\tau_n, \qquad \Delta\tau_n \triangleq -\frac{\phi(\tau_n)}{\omega_0}. \tag{A.3}$$

The above-defined $\Delta\tau_n$ is called *absolute jitter*: It represents the deviation of τ_n from its ideal value and is proportional to the clock's phase error sampled at τ_n [44, 135]. Its importance is due to the fact that in wireline communications it is more useful to represent clock non-idealities in the form of absolute jitter rather than phase error, as typically done for wireless systems [135].

Despite being an inconvenient metric for HSSIs, phase error is commonly reported in the frequency domain in many practical applications, from numerical simulations to laboratory measurements. Following the approach and terminology of [135], the one-sided power spectrum of $v(t)$ is

$$S_v(f) \approx \frac{V_0}{4} S_\phi\left(\left|f - \frac{\omega_0}{2\pi}\right|\right) \qquad \text{for} \quad f \neq \frac{\omega_0}{2\pi}, \tag{A.4}$$

where $S_x(f) \triangleq 2S_{xx}(f)$ for $f \geq 0$ and S_{xx} is the Fourier transform of the time average autocorrelation function of a generic signal $x(t)$.

From such definitions, a normalised power spectrum is defined to be the *jitter density* as

$$J(f) \triangleq \frac{S_\phi(f)}{\omega_0^2} \tag{A.5}$$

due to the fact that it is proportional to the single-sided discrete-time power spectrum of $\Delta\tau_n$ for $f < f_0/2$ (see Equation (25) in [135]). The concept of jitter density is particularly useful when dealing with divided clocks, because it avoids the need for scaling the clock's phase error after every frequency division [135].

When the mean squared value of the absolute jitter is of interest, it can be shown [44, 135] to be given by

$$\sigma^2_{\text{abs}} = \frac{1}{\omega_0^2}\int_0^\infty S_\phi(f)\mathrm{d}f = \int_0^{f_0/2} J(f)\mathrm{d}f. \tag{A.6}$$

Other relevant jitter metrics can be derived depending on the functionalities that the designer needs to check. *Period jitter*, defined as $\Delta\tau_n - \Delta\tau_{n-1}$, is particularly useful to determine the variability of the clock over a single period, which limits the maximum frequency at which digital circuits can operate [44, 135].

Such a concept can be extended to *accumulated jitter*, which represents the variation of clock edges over N periods, i.e $\Delta\tau_n - \Delta\tau_{n-N}$, so that accumulated jitter with $N = 1$ is simply period jitter [44, 135]. Its variance can be expressed as

$$\sigma^2_{\text{acc}} = \int_0^{f_0/2} \left|1 - e^{i2\pi N f/f_0}\right| J(f)\mathrm{d}f \tag{A.7}$$

since, by definition, accumulated jitter over N periods is equivalent to absolute jitter filtered by $1 - z^{-N}$ [135].

As clock is used in HSIOS to retime the data to be transmitted and to sample the received analog waveform, its imperfections are transferred to the data signal itself in the form of *data jitter*. The definitions above hold for this phenomenon as well, the main difference being that the random nature of the data sequence renders data jitter not fully observable (transitions do not occur every T_0) [44].

A.1.2 Jitter Classification

So far, jitter has been presented in general terms, without providing any information regarding what causes it. Such a topic is broad and covers many different mechanisms responsible for jitter generation, from device noise in oscillators to power supply-induced noise in clock distribution networks [13, 44], but this is out of the scope of this thesis. Instead, this section will focus on a general classification based on general characteristics (mainly statistical) of the jitter or of its sources rather than on its precise physical origin.

A conventional classification [44] is summarised in Figure A.1, where the two major categories of interest are:

Random jitter (RJ) is due to fluctuations caused by any type of random process that affects the phase of the clock, mainly device noise (thermal, flicker, etc.). It is characterised by a Gaussian distribution and is therefore *unbounded*.

Deterministic jitter (DJ) is characterised by a bounded probability distribution and is due to deterministic processes such as power supply noise, channel bandwidth limitation (see Section 1.3.2), etc.

Deterministic jitter can be further divided in various sub-categories:

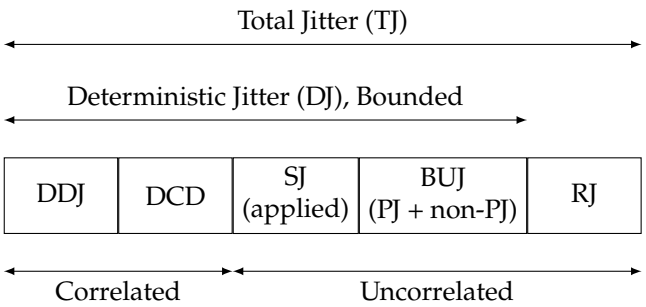


Figure A.1: Jitter classification: DDJ is data-dependent jitter, DCD is duty-cycle distortion, SJ is sinusoidal jitter, BUJ is bounded uncorrelated jitter, PJ is periodic jitter, RJ is random jitter. (Adapted from [44].)

Data-dependent jitter (DDJ) includes all jitter effects which are correlated with the transmitted data, e.g. every different data sequence causes a specific transition to shift in time due to channel bandwidth limitations (see Section 1.3.2).

Duty-cycle distortion (DCD) is due to asymmetries in the clock's duty cycle and is relevant when both clock edges are used (e.g. in half-rate architectures, see Section 1.3.5).

Sinusoidal jitter (SJ) has a sinusoidal time-domain profile and is commonly applied for testing purposes (e.g. CDR characterization, see Section 1.3.1.3).

Bounded uncorrelated jitter (BUJ) is specific to serial digital data, it is bounded but uncorrelated with the transmitted data (e.g. due to crosstalk); it may be further divided into periodic jitter (PJ) and non-periodic jitter.

Moreover, jitter can be classified according to its relation to the transmitted data into *correlated jitter*, comprising DDJ and DCD, and *uncorrelated jitter*, which contains RJ, SJ and BUJ.

Despite the frequency-domain treatment above, knowledge of $J(f)$ is not strictly necessary to infer relevant clock features in terms of jitter [135]. Estimates of σ^2_{abs} are commonly obtained by storing the $\Delta\tau_n$ sequence and plotting its histogram, which approximates the jitter probability distribution, while the various contributions to the total absolute jitter can be extracted and estimated by making realistic assumptions regarding jitter sources and observing the statistical properties of the acquired histogram [44, 135]. In the wireline communications field, such an approach is very useful in the laboratory practice, since oscilloscopes allow so-called *time interval error* (TIE, i.e. the difference between the observed transition in the signal waveform and the instrument's own clock edge) histograms to be taken in order to observe the actual jitter distribution, which is usually non-Gaussian [44, 135].

A.2 Modelling and Simulating CDRs without ISI

In this Section, a model for jitter in CDRs in the case where the incoming signal is not affected by ISI is presented. Despite the fact that ISI can be a major source of

data-dependent jitter (the timing of the signal's edges are modified depending on the transmitted data sequence) and can even amplify the jitter of the transmitted signal [136–139], immediately including this would add a layer of complexity in the model, additionally hindering the identification of other jitter contributions. This notwithstanding, an ISI-less model of jitter in CDRs is still capable of taking into account many relevant jitter sources and their different effect that depend on the chosen CDR architecture, thus providing insight in the phenomenon and aid the system-level design of such circuits.

To verify a model, one could rely on circuit simulators to transmit a random sequence of bits and check the resulting behaviour of the CDR's output. This approach, by considering the actual transistor-level circuit (or HDL code for a digital CDR) provides the most accurate results and is capable of taking into account second-order and non-linear effects, but is characterised by long runtimes and requires an implementation of each architecture to be tested, which is very time-consuming and inefficient. To circumvent such drawbacks, a behavioural, *event-driven* simulator was developed in Octave/MATLAB, so that the different architectures to be considered can be implemented with only minor code modifications, whereas the effort for building the simulator is compensated by its versatility and by the runtime reduction, compared to a circuit simulator. This allows for fast and easy CDR architectural parameter changes to be applied to verify their effect, which is also useful in the first steps of system-level design.

A.2.1 Description of the Event-Driven Simulator

In an event-driven simulator each time step corresponds to the occurrence of an event, i.e. the data and clock transitions in this case, that are computed similarly to what is done in [140] for the simulation of PLLs or [141] for CDRs.

A random sequence of bits is initially generated and associated to a vector containing the transition instants $t_{\mathrm{d}}(\cdot)$ between any two adjacent data bits $d(\cdot)$, as schematically depicted in Figure A.2. In the absence of jitter, such timing samples are simply separated by the inverse of the data rate, T_{B}. When considering a free-running oscillator, a random, normally-distributed (with variance σ) time t^*_{per} is added to each period to model the uncertainty caused by jitter. Instead, when considering a PLL, inverse Fourier transform of its *power spectral density (PSD)* with an additional random phase (uniformly distributed between 0 and 2π) added in each frequency bin is then used to generate the displacements t^*_{abs} so that the n-th bit transition occurs at $nT_{\mathrm{B}} + t^*_{\mathrm{abs}}$.

Two additional vectors, respectively $t_{\mathrm{s,d}}(\cdot)$ and $t_{\mathrm{s,e}}(\cdot)$, are used at the receiver to store the time corresponding to the different clock sampling instants when data and edges are sampled $T_{\mathrm{B}}/2$ seconds apart (this is enough if transmitter and receiver are assumed to be clocked at the same frequency, so that only their phases need to be aligned). As the simulation evolves during time, the n-th elements of data and edge clocks are determined based on the data and edges sampled in the previous instants according to the implemented CDR algorithm, whereas the first element, $t_{\mathrm{s,d}}(0)$, is randomly positioned so to fall inside the first data period.

First, the data and edges samples, respectively forming vectors of elements $d_{\mathrm{s}}(\cdot)$ and $e_{\mathrm{s}}(\cdot)$, are found by looking at the alignment of $t_{\mathrm{s,d}}(\cdot)$ and $t_{\mathrm{s,e}}(\cdot)$ with the timing of the transmitted data $t_{\mathrm{d}}(\cdot)$: As an example, referring to Figure A.2, $e_{\mathrm{s}}(n-1) = d_{\mathrm{s}}(n)$ if $t_{\mathrm{s,e}}(n-1) > t_{\mathrm{d}}(n)$, otherwise $e_{\mathrm{s}}(n-1) = d_{\mathrm{s}}(n-1)$ holds. Note that this implicitly assumes that the channel does not introduce any delay, distortion

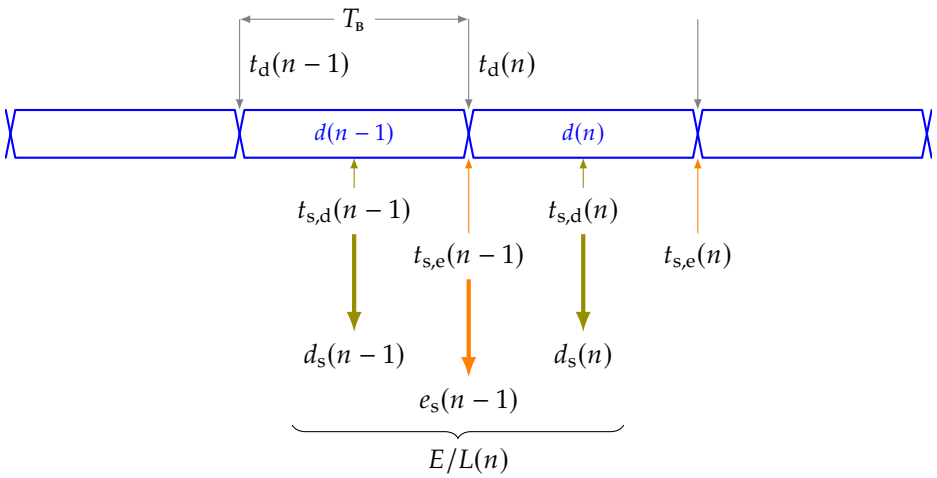


Figure A.2: Timing relations and definition of the main quantities employed to implement the event-driven simulator.

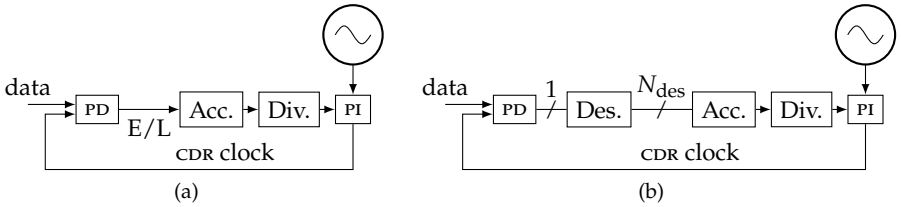
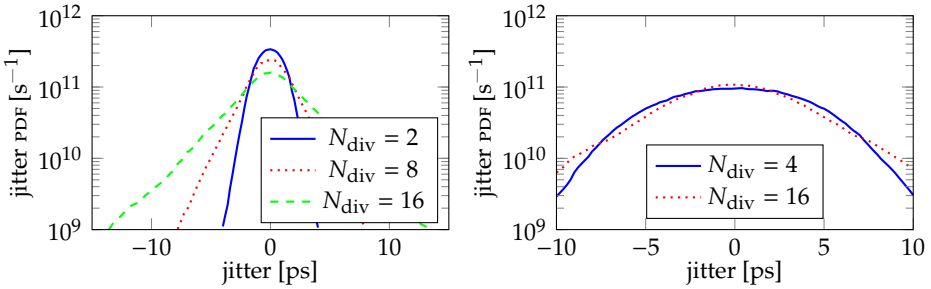


Figure A.3: Simplified CDR schemes considered in this and the following Sections: (a) operates on the incoming serial data, (b) works on data deserialized by a factor N_{des}[1]. The phase detector is followed by a deserializer *Des.* or directly by an accumulator *Acc.*, then a divider *Div.* (i.e. the content of the accumulator is divided by a certain amount and some of its least significant bits are truncated) and the PI, which generates the CDR clock by interpolating multiple clock phases provided by a local oscillator.

or inter-symbol interference, otherwise ISI-induced data-dependent jitter would arise [136] and $t_d(\cdot)$ would not correspond to data transitions at the receiver.

The sampled data and edges $e_s(n-1)$, $d_s(n)$ and $e_s(n)$ are then used to derive the early/late information $E/L(n)$ based on the Alexander algorithm [75] and eventually used to update the PI code in CDRs similar to those of Figure A.3. When early/late detection is performed on the serial data, as in Figure A.3a, they are simply accumulated, scaled by N_{div} (that is usually a power of two for ease of implementation in a digital macro, hence truncating some of the accumulator's least significant bits) and used to drive the PI that has N_{PI} phases. Once the phase of the PI is known, the next data and edge sampling times $t_{s,d}(n+1)$ and $t_{s,e}(n+1)$

[1]Actually, Figure A.3b shows the block diagram of a generic CDR that employs deserialization, but such an implementation would not be useful in many practical cases because phase detection would still take place at data rate. Instead, data and edge samples are separately deserialized and phase detection is performed in the digital macro, as shown in Figure A.8.



(a) $N_{PI} = 64$, $\sigma_{osc,p} = 0.5\,ps$ (b) $N_{PI} = 64$, $N_{des} = 32$, $\sigma_{osc,p} = 0.5\,ps$

Figure A.4: Sample jitter distributions obtained with the event-driven simulator. Plot (a) refers to a CDR fed with serial data with parameters as in Figure A.6, whereas plot (b) considers CDRs on deserialized data as in Figure A.9a.

can be derived: In a nutshell, the phase of the PI determines a delay between $-T_B/2$ and $T_B/2$ in addition to the clock period T_B (special care is needed when switching between phases that would make the delay change from $-T_B/2$ to $T_B/2$ or vice versa). Then the algorithm proceeds for an amount of time steps sufficient to gather enough statistics.

Implementation of CDRs that operate on deserialized data, as that shown in Figure A.3b, is straightforward: The PI code is updated every N_{des} bit periods summing up the early/late values of the last $N_{des} - 1$ bit periods.

The jitter histogram is derived by comparing the transmitted data transitions $t_d(\cdot)$ with the receiver's edge clock $t_{s,e}(\cdot)$. Sample distributions are reported in Figure A.4: The presence of the oscillator noise dominates and makes them fairly Gaussian. Small deviations from Gaussian distributions are observed for small N_{div}, where quantization noise dominates (more on this in the following Sections).

It would be straightforward to apply jitter also to the receiver clock: The distance between two consecutive data (or edge) clock times would not be simply T_B plus the delay associated to the PI, but would also include additional jitter terms as above. The effect on the overall CDR jitter is exactly the same, at least in the cases considered here where there is no channel and thus no amplification of the transmitter jitter.

A.2.2 Analytical Modelling of First-Order CDRs

To derive the analytical formulas for the CDR jitter (t_{CDR}: Difference between the timings of the reconstructed clock and of the received data t_{CK}) the linear model of Figure A.5a is considered as the starting point, assuming a system with a single integrator (i.e. a first-order CDR). As far as only the phase noise of the oscillator is concerned, it results in

$$\frac{\phi_{CDR}}{\phi_{CK}} = \frac{j\omega}{k + j\omega} = \frac{jf/BW}{1 + jf/BW}, \tag{A.8}$$

where $BW = k/(2\pi)$ is the bandwidth and k is a parameter that depends on the system architecture. Figure A.3a considers the case where the early/late (E/L)

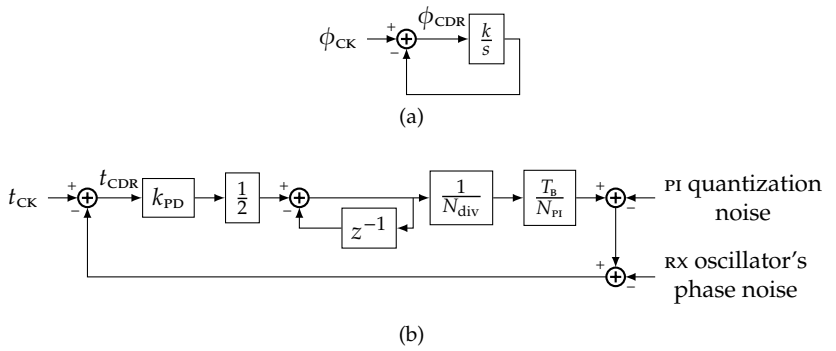


Figure A.5: (a) Laplace-domain linear model of a generic CDR with a single integrator (first order) considering the phase of the transmitter clock as input. (b) Z-domain linear model of an actual CDR (see Figure A.3): The jitter of the transmitter clock (featuring period jitter characterised by $\sigma_{osc,p}$ for a free-running oscillator) is denoted as t_{CK}, while t_{CDR} is the timing error (i.e. jitter) of the reconstructed clock with respect to the received data. Subfigure (b) also indicates the other sources of jitter: The quantization of the PI phases (having RMS value of σ_{quant}) and the noise of the RX oscillator (indistinguishable from the one of the TX).

detection is performed directly on the serial data stream, which will be analysed next.

A.2.2.1 Free-Running Oscillator and E/L Detection on Serial Data

The simplest case concerns E/L detection on the serial stream, with transmitter and receiver clocks generated by free-running oscillators. For simplicity, only the noise of the transmitter's free-running oscillator is considered, and its phase noise power spectral density as a function of the offset frequency f is given by

$$S_{\phi_{CK}\phi_{CK}} = \frac{A}{f^2}, \tag{A.9}$$

where flicker noise is not taken into account to simplify the analysis and obtain close-form expressions. Moreover, the inclusion of flicker noise would also require a second-order CDR to be handled by the receiver.

The period jitter corresponding to Equation A.9 is given by [142]

$$\sigma^2_{osc,p} = \int_0^{+\infty} \left|1 - z^{-1}\right|^2 \frac{T_B}{2\pi}^2 S_{\phi\phi} \mathrm{d}f \approx \frac{AT_B^3}{2}, \tag{A.10}$$

so that A can be determined and substituted into Equation A.9:

$$S_{\phi_{CK}\phi_{CK}} = \frac{2\sigma^2_{osc,p}}{T_B^3 f^2}. \tag{A.11}$$

Since working in the time domain is easier than in the phase domain, consider the block diagram of Figure A.5b corresponding to the architecture in Figure A.5a. If we assume for the TX oscillator the phase noise PSD of Equation A.11, the RMS

value of the absolute jitter of the recovered clock compared to the received signal (i.e. the RMS value of t_{CDR} in Figure A.5b) due to the noise of the oscillator only is:

$$\sigma_{\text{rj}}^2 = \frac{T_{\text{B}}^2}{4\pi^2} \int_0^{+\infty} \frac{2\sigma_{\text{osc,p}}^2}{T_{\text{B}}^3 f^2} \frac{f^2/BW^2}{1+f^2/BW^2} \mathrm{d}f = \frac{\sigma_{\text{osc,p}}^2}{4\pi T_{\text{B}} BW}. \tag{A.12}$$

Notice that Equation A.12 links the *absolute* jitter of the CDR to the *period* jitter of the transmitter clock. As already mentioned, in Figure A.5a, consistently with the event-driven simulator, jitter is applied to the transmitter clock; however, jitter from the oscillator in the receiver has exactly the same effect (as can be seen from where the noise of the RX oscillator is added in the block diagram of Figure A.5b), so that $\sigma_{\text{osc,p}}^2$ in Equation A.12 is indeed the sum of the square of the period jitters of both transmitter and receiver oscillators, which are obviously uncorrelated.

To compute the CDR bandwidth, when it is much smaller than the data rate (which is always the case) the z-domain accumulator can be approximated as

$$\frac{1}{1-z^{-1}} = \frac{1}{1-e^{-sT_{\text{B}}}} \approx \frac{1}{sT_{\text{B}}}. \tag{A.13}$$

The block with gain 1/2 comes from considerations on transition density given that, on average, 0 to 1 or 1 to 0 transitions occur only in half of the bit periods. Comparing Figures A.5a and A.5b, and using Equation A.13, the gain parameter

$$k = \frac{k_{\text{PD}}}{2N_{\text{div}}N_{\text{PI}}} \tag{A.14}$$

can be obtained, which implies that the CDR bandwidth is

$$BW = \frac{k}{2\pi} = \frac{k_{\text{PD}}}{4\pi N_{\text{div}}N_{\text{PI}}}. \tag{A.15}$$

Linearization of the bang-bang *phase detector (PD)* characteristic is discussed in [143–145]. Since σ_{rj} indicates the RMS value of the jitter coming out of the PD, its gain can be written as [146, 147]

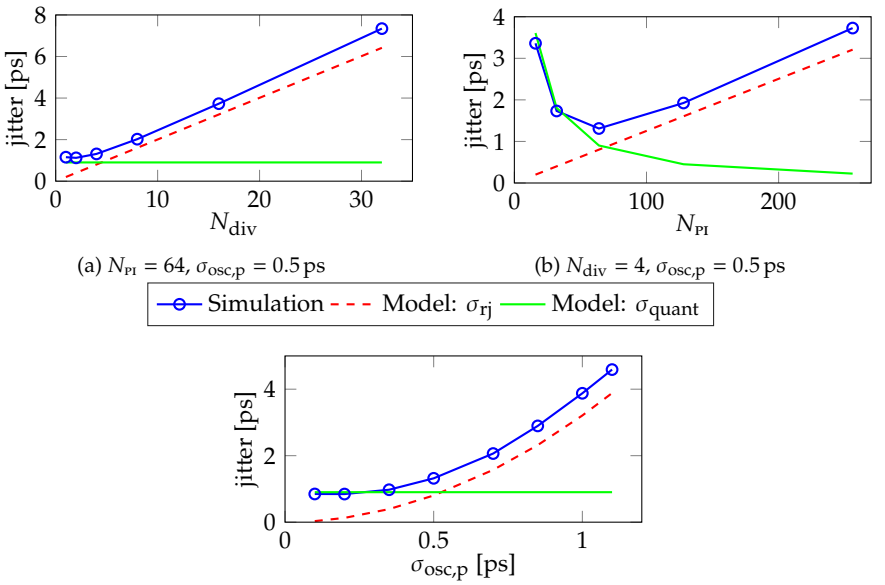
$$k_{\text{PD}} = \frac{2}{\sqrt{2\pi}\sigma_{\text{rj}}}. \tag{A.16}$$

Substituting Equations A.15 and A.16 into Equation A.12, an expression for σ_{rj}^2 is obtained that depends linearly on σ_{rj} itself, eventually yielding

$$\sigma_{\text{rj}} = \frac{\sigma_{\text{osc,p}}^2\sqrt{\pi/2}N_{\text{div}}N_{\text{PI}}}{T_{\text{B}}} \tag{A.17}$$

The results of Equation A.17 are reported by dashed red lines in Figure A.6 and show that the jitter of the clock generators is the main contributor to the CDR jitter when the bandwidth is small (high N_{div} and N_{PI}). On the other hand, for small N_{div} and N_{PI}, Equation A.17 significantly deviates from the results of the event-driven simulations (blue curve), due to the quantization noise associated to the finite number of PI phases [148].

To model this additional contribution, we consider that the desired phase will stay between two of the possible phases provided by the PI. Using time



(a) $N_{\text{PI}} = 64$, $\sigma_{\text{osc,p}} = 0.5\,\text{ps}$

(b) $N_{\text{div}} = 4$, $\sigma_{\text{osc,p}} = 0.5\,\text{ps}$

(c) $N_{\text{PI}} = 64$, $N_{\text{div}} = 4$

Figure A.6: Blue lines with circles: Simulated absolute jitter for a CDR operating at 10 Gb/s on the serial data considering a free-running oscillator as the transmitter clock source. The dashed red and solid green lines refer to Equations A.17 and A.20 respectively. Plots a, b and c show the results as a function of respectively the divider ratio, the number of phases of the PI and the period jitter of the oscillator.

instead of phase, let us say that a delay x with respect to the delay of the n-th phase is desired, whereas the (n+1)-th phase is $\Delta = T_{\text{B}}/N_{\text{PI}}$ away from the n-th phase. Then the system will switch between such two phases since, in a bang-bang implementation, the PD continuously switches between the "late" and "early" states. Thus, the RMS error is

$$\sigma^2(x) = \frac{1}{2}(\Delta - x)^2 + \frac{1}{2}x^2 = \frac{1}{2}\Delta^2 + x^2 - \Delta x. \tag{A.18}$$

In absence of any random jitter, the jitter histogram of the CDR clock t_{CDR} is composed of two Dirac's deltas spaced by $\Delta = T_{\text{B}}/N_{\text{PI}}$, as verified by event-driven simulations (not shown). In the presence of random jitter from the oscillators, x continuously changes over time, so that the average error is

$$\sigma^2_{\text{quant}} = \frac{1}{\Delta}\int_0^{\Delta} \sigma^2(x)\text{d}x = \frac{\Delta^2}{3}. \tag{A.19}$$

In other words, the absolute jitter of the CDR due to quantization of the PI is:

$$\sigma_{\text{quant}} = \frac{1}{\sqrt{3}}\frac{T_{\text{B}}}{N_{\text{PI}}} \tag{A.20}$$

This differs from the formula for PI quantization jitter proposed in [148] (although not derived therein) that features $\sqrt{12}$ instead of $\sqrt{3}$ in Equation A.20. In fact, the

quantization noise of the PI is not the same as in an ADC where an input number or code is approximated to its closest level: Here the PI continuously switches between the two phases that are around the wanted one.

Results of Equation A.20 are reported as green lines in Figure A.6 and reproduce the event-driven simulations in the regions where the jitter induced by the noisy oscillator (as per Equation A.17) is low. Note that the quantization noise of the PI is modelled as a white noise source that must be added to the output of the PI (refer to Figure A.5b) [148]. In principle, Equation A.20 gives the RMS value of this contribution, whose PSD is $2T_B\sigma^2_{\text{quant}}$ (i.e. white over the Nyquist bandwidth), which then goes through the high-pass transfer function of the CDR loop as for Equation A.8. However, since the CDR bandwidth is much lower than the Nyquist frequency $1/(2T_B)$, the integral of the PSD over the squared modulus of the transfer function essentially gives Equation A.20.

Figure A.6a shows that jitter increases with N_{div} (i.e. the CDR bandwidth is reduced) while the dependence on N_{PI} (Figure A.6b) shows that a reduced amount of PI phases available increases the jitter contributed by phase quantization; on the other hand, when N_{PI} increases, the reduction of jitter induced by PI quantization is accompanied by the increase of jitter due to the noisy oscillator (because the CDR bandwidth reduces and thus loses its ability to track the oscillator's jitter). Figure A.6c also shows that the CDR jitter grows quadratically with the oscillator's period jitter (i.e. linearly with the phase noise coefficient A of Equation A.9, consistently with [149]). In Figure A.6 a data rate of 10 Gb/s was considered, but everything scales with T_B so that the agreement between the model's Equations A.17 and A.20, and the event-driven simulation is good in any range of data rates (not shown).

Indications about how to combine the results of Equations A.17 and A.20 and also to include the quantization noise of the bang-bang phase detector are provided in appendix A.2.2.5. Visual observation of the figures in this and following Sections suggests that Equations A.17 and A.20 can be summed together, an evidence that is consistent with [149].

As a final note, it should be mentioned that simulations run with a clock-like data sequence result in a jitter that can be reproduced by a combination of Equation A.20 (because PI quantization jitter is the same) and Equation A.17 divided by two (because the bandwidth is twice as large since we have a 0 to 1 or 1 to 0 transition for each bit period, i.e. transition density equals 1). Simulations with different clock frequencies at transmitter and receiver without phase noise yield a jitter that can be reproduced by Equation A.20 since the phase error between data and clock tends to increase linearly with time and the *desired phase* out of the PI moves from 0 to 2π and experiences quantization due to the PI.

A.2.2.2 Clock from PLL and E/L detection on serial data

The derivation employed in the previous section can be easily adapted to a case where the transmitter is clocked by a PLL instead of a free-running oscillator. To start with, its phase noise PSD can be approximately described by

$$S_{\phi_{CK}\phi_{CK}} = \frac{A}{BW^2_{\text{PLL}} + f^2}, \tag{A.21}$$

i.e. a flat PSD up to the PLL bandwidth BW_{PLL} followed by a $1/f^2$ slope. In this case the *absolute* jitter is given by:

$$\sigma^2_{\text{PLL}} = \int_0^{+\infty} \left(\frac{T_{\text{B}}}{2\pi}\right)^2 S_{\phi\phi} \mathrm{d}f \approx \frac{AT_{\text{B}}^2}{8\pi BW_{\text{PLL}}} \tag{A.22}$$

This analysis starts with Equation A.8 as well, the phase PSD of Equation A.21 is used instead of the one from Equation A.12, yielding

$$\sigma^2_{\text{rj}} = \sigma^2_{\text{PLL}} \frac{1}{1 + \frac{1}{BW_{\text{PLL}}\sqrt{\pi/2}} \sigma_{\text{rj}} 4\pi N_{\text{div}} N_{\text{PI}}}, \tag{A.23}$$

where Equations A.15 and A.16 were used for the CDR bandwidth. Equation A.23 can then be solved to get:

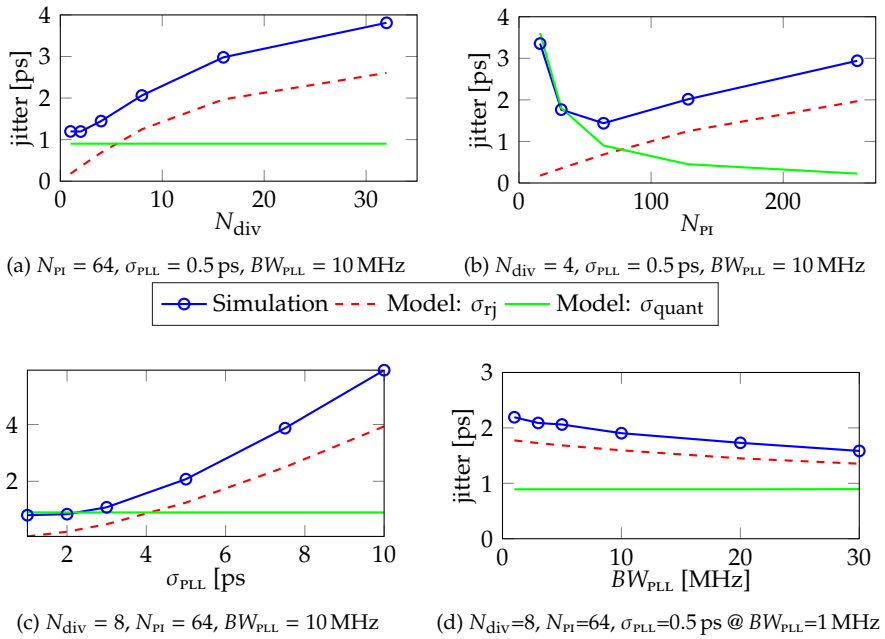
$$\begin{aligned} \sigma_{\text{rj}} &= -\frac{1}{C} + \sqrt{\frac{1}{C^2} + \sigma^2_{\text{PLL}}} \\ C &= BW_{\text{PLL}} 8\pi\sqrt{\pi/2} N_{\text{div}} N_{\text{PI}} \end{aligned} \tag{A.24}$$

Results of Equation A.24 are reported by dashed red lines in Figure A.7 and compared with the event-driven simulator. For small N_{div} and N_{PI} the event-driven simulations tend to reproduce the jitter provided by Equation A.20 (PI quantization). The trends of CDR jitter vs. division ratio (Figure A.7a), number of PI phases (Figure A.7b) and clock jitter (Figure A.7c) are analogous to those shown in Figure A.6. Regarding the effect of the PLL bandwidth, in Figure A.7d BW_{PLL} is varied and, simultaneously, σ^2_{PLL} is decreased as $1/BW_{\text{PLL}}$, so to describe a situation where the VCO of the PLL is the same, but the increased PLL bandwidth better filters out its phase noise at low frequencies. In other words, a phase-noise PSD as in Equation A.21, where A is constant, is considered; then, since the transfer function between the clock jitter and the CDR jitter (Equation A.8) is high-pass, increasing the PLL bandwidth has a minor effect on reducing the overall absolute jitter of the system since it only affects the low frequency portion of the clock phase noise PSD.

Since the results obtained with the PLL are in line with the ones obtained considering a free-running oscillator, apart from more involved formulas, only the latter will be considered for simplicity in the following (although PLL clock synthesis could be included also in the more complex situations described in the next Sections).

A.2.2.3 Free-Running Oscillator and E/L Detection After Deserialization

Consider now a CDR systems where the E/L detection is performed after deserialization. Two different architectures are shown in Figure A.8: In the case shown in Figure A.8a the E/L read from the deserialized bits are added together, in this way losing almost no information compared to voting where a single E/L is derived from the parallel E/L vector instead (shown in Figure A.8b). However, even with the adder, some information is lost: E/L detection cannot be performed on the last bit of the parallel stream since this requires the first data sample of the next parallel word; for this reason, in Figure A.8a the vector coming out of the E/L



(a) $N_{PI} = 64$, $\sigma_{PLL} = 0.5$ ps, $BW_{PLL} = 10$ MHz (b) $N_{div} = 4$, $\sigma_{PLL} = 0.5$ ps, $BW_{PLL} = 10$ MHz

(c) $N_{div} = 8$, $N_{PI} = 64$, $BW_{PLL} = 10$ MHz (d) N_{div}=8, N_{PI}=64, σ_{PLL}=0.5 ps @ BW_{PLL}=1 MHz

Figure A.7: Blue lines with circles: Simulated absolute jitter for a CDR operating at 10 Gb/s on the serial data considering a PLL in the transmitter. The dashed red and solid green lines refer to Equations A.24 and A.20, respectively. Plots a,b,c and d show the results as a function of respectively the divider ratio, the number of phases of the PI, the absolute jitter of the PLL and the bandwidth of the PLL. In plot d, σ_{PLL} is scaled as $1/BW_{PLL}$).

block is $N_{des} - 1$ bit wide. Additionally, when the CDR works on deserialized data, the PI code is updated only after N_{des} bit periods.

By performing event-driven simulations without transmitter jitter, we notice that the phase takes more than two values. In fact, the input of the PI is updated by a value that is the output of the adder after division. If the division ratio is small and a sufficient majority of Early or Late is detected at the same time from the deserialized data, the PI phase moves by more than one step. The number of lines in the jitter histogram is approximately N_{des}/N_{div} (not shown). This is different from the cases analysed in the previous subsections where only one early/late is added to the accumulator and thus to the PI input.

Sample simulation results including oscillator jitter are reported in Figure A.9a. We see that for small CDR bandwidths (high N_{div}) the jitter is almost the same, whereas working with deserialized data is detrimental in the cases where quantization of the PI dominates (small N_{div}). The results in Figure A.9a suggest that Equation A.17 can still be used Equation A.20 has to be modified. The fact that given the target phase, the PI switches between more than two phases, suggests the following modification, so that the former expression (Equation A.20) is recovered in the cases when $N_{des} < N_{div}$:

$$\sigma_{quant} = \frac{1}{\sqrt{3}} \frac{T_B}{N_{PI}} max\left(1,\ floor\left(\frac{N_{des}}{2N_{div}}\right)\right). \tag{A.25}$$

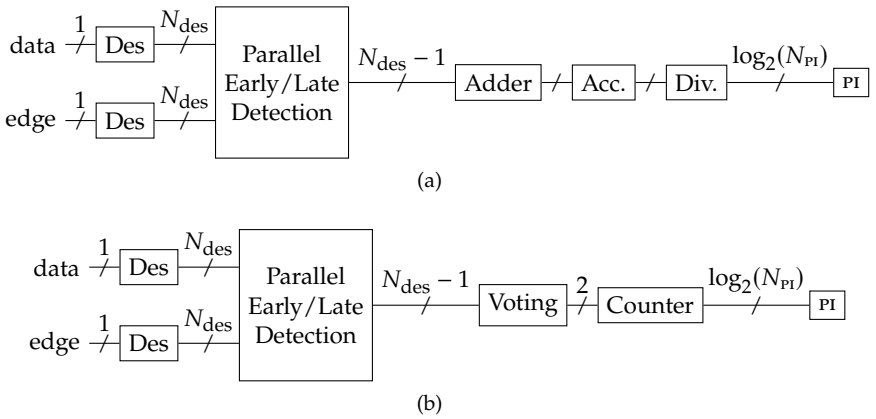


Figure A.8: Example of CDR architectures performing the E/L detection on deserialized data and edge samples, as two possible implementations of Figure A.3b. In both cases the PI code is updated after N_{des} data periods. In plot (a) the E/L resulting from the parallel data are summed together (with sign), then accumulated and divided before driving the PI. The voting in plot (b), instead, produces "up" and "down" signals for the counter driving the PI.

The results of Equation A.25 are compared with event-driven simulations in Figure A.9b for situations with small N_{div}, so that the jitter contribution due to the oscillator noise is negligible. Instead, Figure A.9c plots the results of Equations A.17 and A.25 vs. event-driven simulations for E/L detection after 32-bit deserialization, demonstrating that quantization noise due to discrete PI phases dominates at small N_{div} (high bandwidth), while noise of the oscillator dominates for high N_{div} (small bandwidth). This leads to a trade-off in the choice of the divider ratio, with the optimum division ratio that depends on N_{des} and on the jitter of the oscillators (compare Figures A.9c and A.9d).

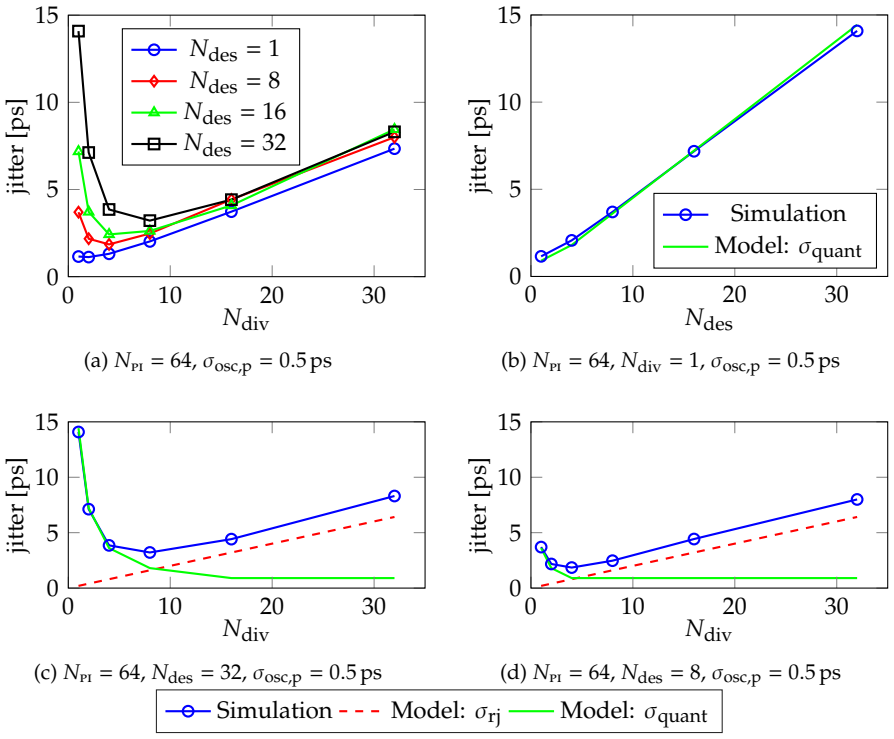
A.2.2.4 Free-Running Oscillator and E/L Detection After Deserialization with Voting

An alternative scheme for E/L detection on deserialized data and edges is depicted in Figure A.8b. In this case, majority voting is performed on the parallel E/L vector: The output of the voting block controls the up and down inputs of a counter that drives the PI. This guarantees that the phase is always updated by ±1 only. As a result, the noise related to quantization of the PI is the same as when E/L detection is performed on serial data, i.e. Equation A.20 holds.

On the other hand, the voting reduces the bandwidth of the CDR, meaning that Equation A.17 (jitter due to the noise of the transmitter oscillator) has to be modified as follows:

$$\sigma_{rj} = \frac{\sigma^2_{osc,p}\sqrt{\pi/2}N_{des}N_{PI}}{2T_B}. \tag{A.26}$$

In other words, the bandwidth is reduced by a factor $N_{des}/2$ (there are on average $N_{des}/2$ useful transitions in a deserialized vector, but the voting counts only one) compared to the CDR working on serial data (where half of the bit periods show useful transitions). Figure A.10 compares the model consisting of Equations A.26



(a) $N_{\text{PI}} = 64$, $\sigma_{\text{osc,p}} = 0.5\,\text{ps}$

(b) $N_{\text{PI}} = 64$, $N_{\text{div}} = 1$, $\sigma_{\text{osc,p}} = 0.5\,\text{ps}$

(c) $N_{\text{PI}} = 64$, $N_{\text{des}} = 32$, $\sigma_{\text{osc,p}} = 0.5\,\text{ps}$

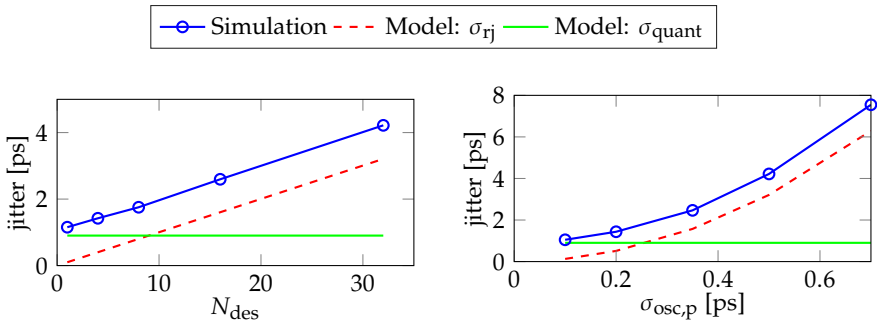
(d) $N_{\text{PI}} = 64$, $N_{\text{des}} = 8$, $\sigma_{\text{osc,p}} = 0.5\,\text{ps}$

Figure A.9: (a) Results of event-driven simulations for the scheme in Figure A.8a considering a data-rate of 10 Gb/s and a free-running oscillator in the transmitter. Plots b, c and d: Blue lines represent the simulated absolute jitter, the red and green lines refer to Equations A.17 and A.25, respectively. Plot b is for varying N_{des} (parallel bits), while plots c and d are for varying N_{div} at fixed N_{des}.

and A.20 with event-driven simulations. In this system architecture, deserialization leads to an increase in the impact of the phase noise of the oscillators on the overall CDR jitter, while the effect of PI quantization is limited.

Another important aspect related to the digital implementation of the CDR is the latency of the loop [148, 150, 151]. In fact, E/L detections as well as voting require some clock cycles, denoted as N_{del} at the frequency of the deserialized data, so that the associated delay is $N_{\text{des}}N_{\text{del}}T_{\text{B}}$ when expressed in terms of the data rate. This delay has an effect on the loop bandwidth (although small, as will be shown later) and more relevantly on the noise associated to PI quantization: In fact, the outcome of the E/L detection is delayed before reaching the PI so that the output phase features limit cycles including more than two phases, due to outdated control words computed previously. Simulations using the event-driven simulator when the update of the PI code is delayed are reported by blue lines in Figure A.11. Even when the jitter of the clock is small, the jitter of the CDR increases linearly with N_{del}. In particular, the jitter contribution due to quantization can be modelled as:

$$\sigma_{\text{quant}} = \frac{1}{\sqrt{3}} \frac{T_{\text{B}}}{N_{\text{PI}}} (1 + N_{\text{del}}). \tag{A.27}$$



(a) $N_{PI} = 64$, $\sigma_{osc,p} = 0.5\,ps$ (b) $N_{PI} = 64$, $N_{des} = 32$

Figure A.10: Blue lines: Simulated absolute jitter for a CDR operating at 10 Gb/s on the deserialized data (using the scheme in Figure A.8b) considering a free-running oscillator in the transmitter. The red and green lines refer to Equations A.26 and A.20, respectively. Plot a is for varying N_{des} (parallel bits) while plot b is for varying period jitter of the transmitter oscillator.

In fact, the state of the CDR evolves in limit cycles whose amplitude is proportional to N_{del} [152]. As a consequence, the output of the PI switches among a larger set of phase values centred on the phase to be tracked, thus increasing the quantization noise proportionately to N_{del}. Therefore, A.27 should be interpreted as the quantization noise due to the intrinsic one-cycle delay of the CDR plus another quantization noise contribution proportional to N_{del}.

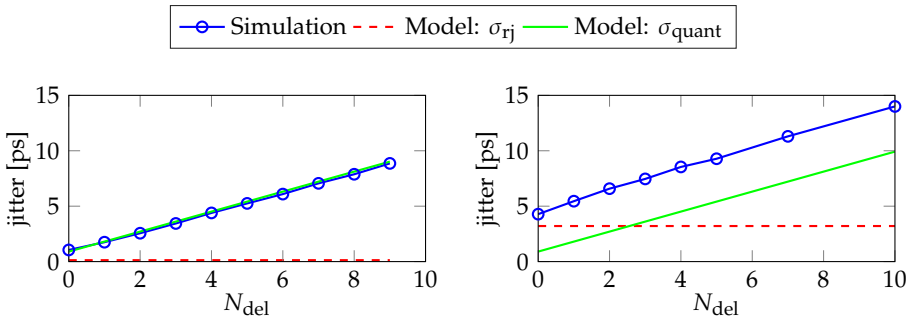
On the other hand, Equations A.26 and A.27 clearly reproduce quite accurately the jitter of the event-driven simulator, suggesting that Equation A.26 is still valid, i.e. that the additional $z^{-N_{des}N_{del}}$ term in the loop transfer function has a small impact on transforming the clock jitter into CDR jitter. Notice that a latency of approximately $0.5N_{des}T_B$ is in any case present when working with deserialized data even if the digital core latency is not included.[2]

The effect of digital core latency on the CDR jitter is severe and mandates inclusion of a divider after the counter in Figure A.8b. This reduces the noise associated with quantization by roughly N_{div}, but unfortunately reduces the bandwidth by the same amount, hence increasing the CDR jitter due to clock jitter.

A.2.2.5 Combining the Different Jitter Sources

Formulas for the CDR jitter due to noisy oscillators (σ_{rj}) and finite PI phases (σ_{quant}) were provided in the Sections above: The two contributions have been derived independently, but since k_{PD} depends on the jitter itself, they are interdependent. Furthermore, the quantization noise due to the binary output of the bang-bang phase detector [146] was neglected. For simplicity, when considering a CDR operating on serial data with a free-running oscillator (same as appendix A.2.2.1), its

[2]This stems from the fact that some time is necessarily required for the gathered information on transitions to be processed and applied: In fact, early/late detection is performed after deserialization (see Figure A.8), so that all the N_{des} data have to be acquired before the detection actually starts. Such a latency depends on the position of the data/edge in the deserialized vector, so that $0.5N_{des}T_B$ represent the average delay over the whole vector.



(a) $N_{PI} = 64$, $N_{des} = 32$, $\sigma_{osc,p} = 0.1\,ps$ (b) $N_{PI} = 64$, $N_{des} = 32$, $\sigma_{osc,p} = 0.5\,ps$

Figure A.11: Blue lines: Simulated absolute jitter for a CDR operating at 10 Gb/s on the deserialized data (using the scheme in Figure A.8b) considering a free-running oscillator in the transmitter. The red and green lines refer to Equations A.26 and A.27, respectively. Plots a and b consider two different values of the period jitter of the clock and plot the CDR jitter for varying N_{del} (delay of the digital core).

total jitter can be expressed as

$$\sigma^2_{CDR,tot} = \sigma^2_{CDR}\Big|_{osc} + \sigma^2_{quant} + \sigma^2_{qPD}, \tag{A.28}$$

where the first term is the contribution to the CDR jitter of the oscillator period jitter. This can be computed as in appendix A.2.2.1 but considering that k_{PD} would depend on $\sigma_{CDR,tot}$ instead of σ_{rj}. So, adapting Equation A.12 one finds

$$\sigma^2_{CDR}\Big|_{osc} = \frac{\sigma^2_{osc,p}}{4\pi T_B BW} = \frac{\sigma^2_{osc,p}\sqrt{\pi/2}N_{div}N_{PI}}{T_B}\sigma_{CDR,tot} = \sigma_{rj}\sigma_{CDR,tot}, \tag{A.29}$$

where σ_{rj} is given by Equation A.17.

The term σ^2_{quant} in Equation A.28 is the square of what is expressed in Equation A.20, while σ^2_{qPD} is the contribution of the quantization of the PD output to the CDR jitter, which can be modelled as white noise at the input of the accumulator with RMS value $0.5(1-1/\pi)$ [146]. This corresponds to a white PSD on the Nyquist band from 0 to $1/(2T_B)$ filtered by the CDR loop. Therefore,

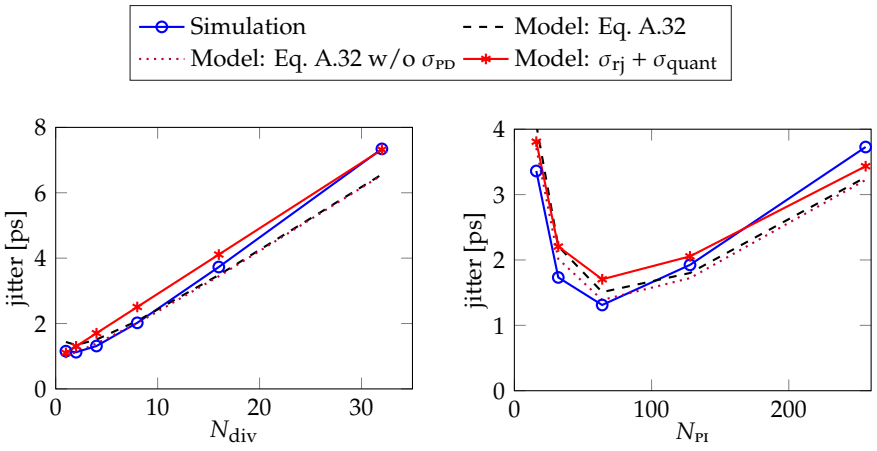
$$\sigma^2_{qPD} = \frac{1}{2}\left(1-\frac{1}{\pi}\right)2T_B\frac{4}{k^2_{PD}}\int_0^{+\infty}\frac{BW^2}{BW^2+f^2}df = \sigma_{PD}\sigma_{CDR,tot} \tag{A.30}$$

can be computed by introducing the term

$$\sigma_{PD} = \left(1-\frac{1}{\pi}\right)\frac{T_B}{2N_{div}N_{PI}}\sqrt{\frac{\pi}{2}}. \tag{A.31}$$

By inserting Equations A.29 and A.30 in Equation A.28, one gets a second order equation yielding

$$\sigma_{CDR,tot} = \frac{1}{2}\left(\sigma_{rj} + \sigma_{PD} + \sqrt{(\sigma_{rj}+\sigma_{PD})^2 + 4\sigma^2_{quant}}\right). \tag{A.32}$$



(a) $N_{PI} = 64$, $\sigma_{osc,p} = 0.5\,ps$ (b) $N_{div} = 4$, $\sigma_{osc,p} = 0.5\,ps$

Figure A.12: Blue lines: Simulated absolute jitter for a CDR operating at 10 Gb/s on the serial data (same as in Figures A.6a and A.6b). The results of Equation A.32 with and without the term σ_{PD} are compared with the sum of σ_{rj} and σ_{quant} from Equations A.17 and A.20.

The results of Equation A.32 are compared with the event-driven simulations in Figure A.12. Comparison between black and magenta lines points out that σ_{PD} has a minor effect on the CDR jitter, consistently with [148]. Overall Equation A.32 matches the event/driven simulations, with some deviation at high division ratios or high number of PI phases. On the other hand, the event-driven simulations are matched quite well also by the simple sum $\sigma_{rj} + \sigma_{quant}$, an observation consistent with [149] which can be verified by visual inspection of the other figures in the previous Sections.

A.2.3 Comparison with Circuit-Level Simulations

To show the applicability of the models proposed in the previous section to realistic situations and validate them against schematic-level simulations, the digital CDR circuit of [109] was simulated using the mixed-signal simulator *XA-VCS*. E/L detection is performed after 1:40 deserialization with majority voting, such a result is deserialized 1:2 and a second majority voting is performed; the resulting signal changes the phase of the PI by -1,0 or 1.

Results are reported in Figure A.13. The mixed-signal environment couples the gate-level analog circuit to the HDL description of the digital macro. The oscillators and their noise are not included at schematic level: Instead, square waves with jittered edges are fed into the CDR to implement period jitter with RMS value $\sigma_{osc,p}$. Due to the non-linearity of the PI, different delays between TX and RX clocks are considered in order to span different portions of the PI characteristic and average out the effect of such non-linearity. As a result, different red circles are shown in Figure A.13 at the same $\sigma_{osc,p}$ value, so that the cloud composed of those matches quite well the results of the event-driven simulator (that for this analysis has been slightly modified to handle the double voting algorithm of the CDR considered here).

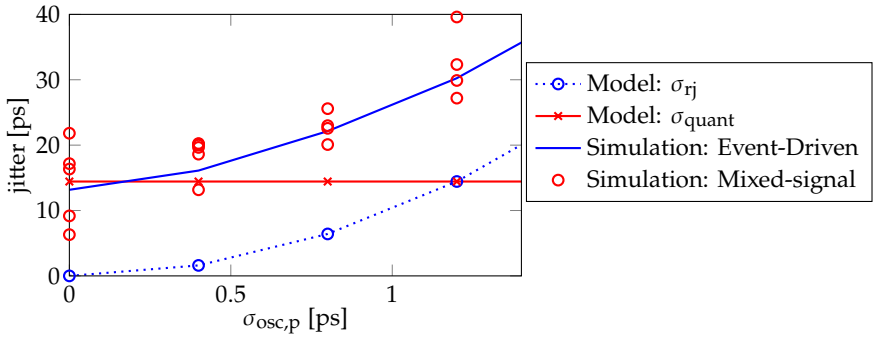


Figure A.13: Simulated jitter of the CDR described in [109] operating at 6.25 Gb/s. Mixed-signal simulations are compared with the event-driven simulator and with the model in appendix A.2.2.4.

Figure A.13 also reports the results of the compact formulas proposed in appendix A.2.2.4. Since a 1:2 deserialization is performed after the first voting on the 1:40 deserialized data and edge, in Equation A.26 $N_{des} = 80$ was set. Concerning the value of N_{del} to be inserted into Equation A.27, the delay of the digital part of the CDR is about 8 cycles of the clock that times the 1:40 deserialized data (of period $N_{des}T_B = 40T_B$). Since the final voting is essentially on 80 bits, $N_{del} = 4$ has been inserted into Equation A.27. Notice that Equation A.27 gives a rather fine estimate of the CDR jitter due to the limit cycle of the PI phases (i.e. at $\sigma_{osc,p} = 0$ in Figure A.13), while Equation A.26 captures the increment of the CDR jitter when $\sigma_{osc,p}$ increases.

A.2.4 Validity of the Model in Second-Order CDRs

Although the analysis carried out in the previous Section considered first-order CDRs only, working with real-world applications requires to study frequency tracking in order to consider cases when transmitter and receiver are not clocked at precisely the same frequency (indeed, differences in the order of tens or hundreds of ppm can be expected in real applications [34, 153]). To this end, it was straightforward to add an additional integrator in the loop and verify that the jitter is in most cases rather similar in first- and second-order CDRs. As will be shown in the following, this is particularly true when the integral path in the second-order CDR is strong enough to compensate for frequency differences between transmitter and receiver oscillators as large as 100 ppm, but not so strong as to affect the loop transfer function experienced by noise. An integral path stronger than needed may affect the CDR transfer function, thus modifying the output jitter due to the noisy oscillators and even lead to loop stability issues. On the other hand, when the jitter due to PI quantization is dominant, the effect of the CDR transfer function is negligible (the quantization noise is white up to the Nyquist frequency, which is orders of magnitude higher than the CDR bandwidth) and first- and second-order CDRs display essentially the same jitter, so the expressions for σ_{quant} reported in the previous Section apply to both cases.

A compact formula for σ_{RJ} can be obtained only in the simplified case of CDRs performing E/L detection on serial data and featuring a free-running oscillator (i.e. the case analysed in appendix A.2.2.1). To derive this expression, consider for

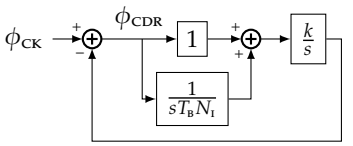


Figure A.14: Simplified second-order CDR scheme as considered in this Section, based on the one for first-order architectures shown in Figure A.5a. Notice that the gain of the proportional path takes the value $k_P = k = k_{PD}/(2N_{div}N_{PI})$ from Equation A.14

the second-order CDR the scheme of Figure A.14, updated from Figure A.5a, which shows an additional block before the accumulator, featuring a direct path with unitary gain (i.e. the gain of the proportional path k_P will be the factor $k = \frac{k_{PD}}{2N_{div}N_{PI}}$ from Equation A.14), in parallel with an accumulator followed by a division factor N_I (i.e. the gain of the integral path is $k_I = k/T_B N_I$, with the notation used in the first-order CDR). In this structure, Equation A.8 is modified as

$$\frac{\phi_{CDR}}{\phi_{CK}} = \frac{(j\omega)^2}{(j\omega)^2 + j\omega k_P + k_I}, \tag{A.33}$$

so that the jitter transfer function to be inserted into Equation A.12 is given by

$$|H(f)|^2 = \frac{f^4}{f^4 + \left(\frac{k^2}{4\pi^2} - \frac{2k}{4\pi^2 T_B N_I}\right) f^2 + \left(\frac{k}{4\pi^2 T_B N_I}\right)^2} = \frac{f^4}{(f^2 + f_1^2)(f^2 + f_2^2)} \tag{A.34}$$

$$= \frac{f^2}{f_2^2 - f_1^2}\left[\frac{1}{1 + \left(\frac{f}{f_2}\right)^2} - \frac{1}{1 + \left(\frac{f}{f_1}\right)^2}\right], \tag{A.35}$$

where the expressions for f_1^2 and f_2^2 are omitted.

By inserting Equation A.35 into Equation A.12 in place of $f^2/(BW^2 + f^2)$ one obtains the integral

$$\sigma_{RJ}^2 = \left(\frac{T_B}{2\pi}\right)^2 \int_0^{+\infty} \frac{2\sigma^2}{T_B^3 f^2} \frac{f^2}{f_2^2 - f_1^2}\left[\frac{1}{1 + \left(\frac{f}{f_2}\right)^2} - \frac{1}{1 + \left(\frac{f}{f_1}\right)^2}\right] df$$

$$= \frac{\sigma^2}{2\pi^2 T_B} \frac{1}{f_2^2 - f_1^2} \frac{\pi}{2}(f_2 - f_1) = \frac{\sigma^2}{4\pi T_B} \frac{1}{f_2 + f_1}, \tag{A.36}$$

based on the known identity $\int_0^{+\infty} \frac{1}{1+x^2} dx = \frac{\pi}{2}$. By equating the denominators of Equation A.34 one finds that, when both f_1^2 and f_2^2 are real (i.e. $k > 4/T_B N_I$), $f_1 + f_2 = \frac{k}{2\pi}$ holds and the random jitter variance is thus given by

$$\sigma_{RJ}^2 = \frac{\sigma^2}{2T_B k}. \tag{A.37}$$

Surprisingly, the integral gives exactly the same result as Equation A.12, that then leads to Equation A.17, with no dependence on the integral path gain. In other

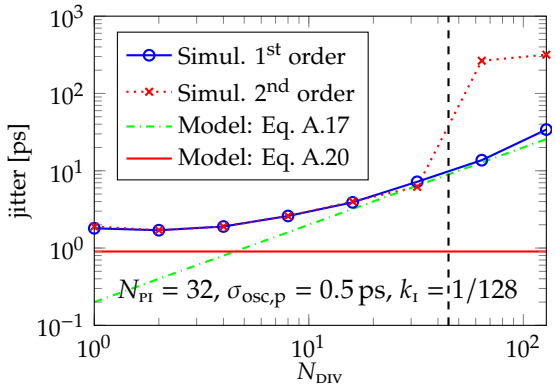


Figure A.15: Absolute jitter of first- and second-order CDRs considering free-running oscillators and different division ratios. Symbols correspond to the event-driven simulator without ISI of Appendix A.2.1, while lines without markers are the equations of appendix A.2.2.1 for a first-order CDR. The vertical line displays the value of N_{DIV} corresponding to the condition $k = 4/T_B N_I$.

words, if $k > 4/T_B N_I$ holds, the second-order CDR has exactly the same jitter as the first-order one when working on data clocked by a free-running oscillator. This is verified by the results in Figure A.15: The event-driven simulator without ISI gives exactly the same results in terms of jitter as far as $k > 4/T_B N_I$ (vertical dashed line) holds. In that range the jitter is also well described by Equation A.17. In addition, making the CDR operate in that regime is not a good choice since it corresponds to poles with imaginary part that result in overshooting.

A.3 Modelling and Simulating CDRs with ISI

As mentioned at the beginning of the previous section, ISI can be a major source of data-dependent jitter and can amplify the jitter introduced by the transmitter, hence increasing its effect at the receiver. Therefore, despite the useful results on the CDR performance that can be obtained without considering ISI, accounting for it is of paramount importance when dealing with modern wireline transceiver standards that target communication at high data rate over high-loss channels that introduce relevant ISI.

A.3.1 Description of the Simulator

In order to take ISI into account, a simulator was developed whose working principle is very similar to the one without ISI detailed in Appendix A.2.1, despite the major difference that a full analog waveform $v(t)$ based on the transmitted data sequence has to be generated, as shown in the scheme of Figure A.16. This is accomplished by simply combining the target channel's pulse response (computed with the method presented in Section 2.2.3, possibly taking into account also the receiver's AFE) with the transmitted bit sequence. Notice that introducing transmit-side jitter when considering ISI mandates the use of the channel's step response, which is simply computed by integration of the pulse response; this is

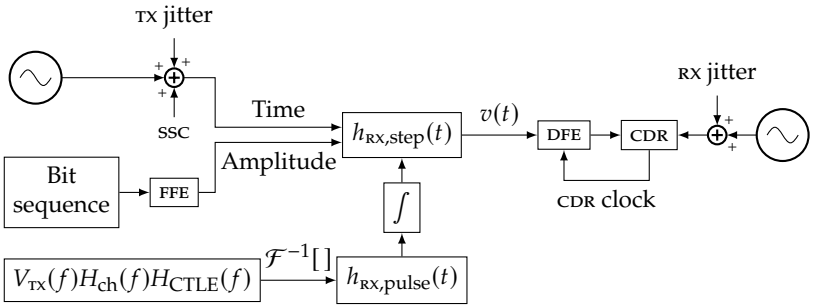


Figure A.16: Flowchart of the CDR simulator with ISI; refer to Section 2.2.1 for details on $V_{TX}(f)H_{ch}(f)H_{CTLE}(f)$.

necessary because the application of transmitter jitter would require "stretching" in time the pulse response for each transmitted bit depending on the jitter values at the beginning and at the end of the pulse, whereas working with step responses simplifies the computation because all that is needed is to shift in time the application of the step without affecting its shape. The same principle is exploited to apply arbitrary clock modulations, typically referred to as *spread-spectrum clocking*, used to scatter the clock's energy content in a larger range of frequencies, hence reducing the severity of the induced electromagnetic interference.

More specifically, transmitter's jitter and spread-spectrum modulation are superposed to the ideal generated clock to obtain a signal with the desired features to provide timing to the transmitter. Meanwhile, the data sequence to be transmitted (e.g. a PRBS, a clock-like pattern or a completely random sequence) is generated and FFE is applied (if enabled) to the symbol values, so that the PAM-2's '+1' and '-1' levels split depending on the data pattern and applied equalization. The resulting data values are then used as amplitudes of the step responses that are used to generate the analog waveform for the receiver (recall that the step response embeds the effects of the receiver's AFE), whereas the jittered clock sequence determines the application instants of such steps.

At the receive side, the basic mechanism behind the simulator with ISI is retained from the one without ISI presented in Appendix A.2.1, and the relations among its main quantities are schematically depicted in Figure A.17. The analog waveform $v(t)$, representing the received signal after the AFE, is composed of samples spaced by δt, which are interpolated to obtain the precise voltage value corresponding to the (possibly jittered) sampling clock instants that may occur in between; such analog sampled values are then equalized with DFE before slicing, so that the DFE correction does not have to be applied to the full analog waveform, thus simplifying computations. This procedure is performed for both data and edge samples, which are then fed to an Alexander phase detector to determine early/late relationships and subsequently processed according to the chosen CDR architecture; additionaly, the instants corresponding to the zero-crossings of $v(t)$ are stored for possible further processing and are indicated as $t_{0\text{-}X}$.

The CDR topologies implemented in Appendix A.2 (e.g. those shown in Figure A.3) and the second-order one of Appendix A.2.4 are also supported in the simulator with ISI.

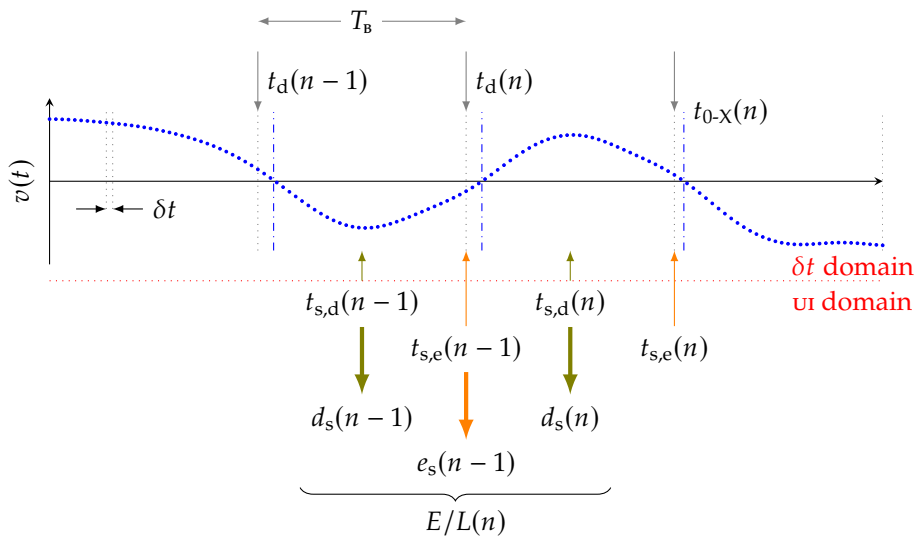


Figure A.17: Timing relations and definition of the main quantities employed to implement the event-driven simulator with ISI. With respect to Figure A.2, notice $v(t)$ representing the received analog waveform composed of samples spaced by δt and the domain of the CDR where all vectors contain samples separated by 1 UI (plus jitter contribution); $t_{0\text{-}X}$ indicates $v(t)$'s zero-crossings.

A.3.1.1 On the Estimation of Jitter in the Presence of ISI

In Appendix A.2 jitter was evaluated by simply computing the difference between the data transition instant given by the transmitter's clock and the sampling instant of the receiver's clock. However, the channel's ISI introduces data-dependent jitter that alters the (previously constant) relation between the transmitter's clock and the transitions of the received signal, and its propagation delay adds a constant offset between the two, hence mandating a reliable method to extract jitter information without neglecting any contribution that may affect operations of the CDR.

Data-dependent jitter alone can be computed, in the absence of transmit-side jitter, by simply considering a horizontal cut at 0 V (or at the decision thresholds for a generic PAM-N signalling scheme) of the eye diagram obtained with the statistical method proposed in Chapter 2, because the distribution of the signal crossings in presence of ISI is indeed the jitter introduced by the channel. Instead, when jittered data is transmitted, the time difference between signal transitions at the transmitter (which are known) and at the receiver ($t_{0\text{-}X}$, which have to be searched in the analog waveform) can be used to extract information on the channel-induced data-dependent jitter.

However, measuring the overall jitter when considering all possible contributors and the CDR is not so straightforward. By computing the difference between the actual signal transitions $t_{0\text{-}X}$ and the timing of the edge samples at the receiver $t_{s,e}$, a measure of how well the CDR tracks the threshold crossings of the incoming signal is obtained, which can be used to determine the amount of jitter affecting the phase detector; with this approach, measuring null jitter would mean that the CDR tracks every transition (of course, this cannot happen, as the minimum

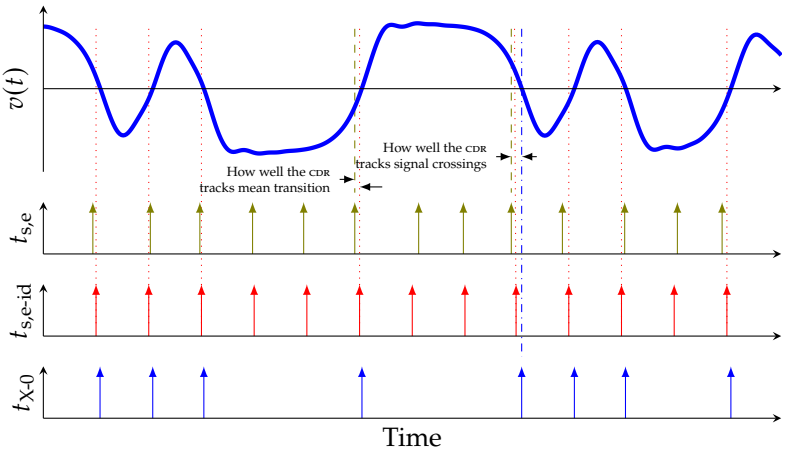


Figure A.18: Estimating jitter when the input signal is affected by ISI: $v(t)$ is the received analog waveform, $t_{s,e}$ shows the sampling instants of the instantiated CDR architecture, $t_{s,e\text{-id}}$ is a quasi-ideal linear CDR that can be used as golden reference to track the mean signal transition and $t_{0\text{-}X}$ represents the zero-crossing instants of $v(t)$; their timings are indicated respectively by dashed green, dotted red and dash-dotted blue lines.

latency of one cycle would require the CDR to "foresee" each successive phase error). Alternatively, a quasi-ideal, linear CDR can be instantiated in parallel to the CDR under test so that the former behaves as a golden reference that tracks the mean crossing of the received signal (instead of the actual transitions of the method mentioned previously), and jitter is evaluated as the difference between the quasi-ideal sampling instants $t_{s,e\text{-id}}$ and those generated by the CDR architecture under investigation; in this way, measuring null jitter means that the CDR is locked to the average signal transition (or, at least, up to how well the reference CDR can achieve this). Figure A.18 depicts schematically such options available for jitter estimation in the presence of ISI, which are also summarised in Table A.1.

A.3.2 Testing the Effect of ISI on CDRs

In order to assess the validity of the simulator with ISI, the phenomenon of jitter amplification in lossy channels, where jitter applied to the transmitted data is amplified by ISI, was tested. This was investigated in [138] considering clock channels in the frequency domain by assuming that the periodicity of the clock sequence eliminates jitter contributions due to pure ISI, so that jitter at the channel output is caused only by transmit jitter. Considering an approximate loss model described by the channel transfer function $H(\omega) = e^{-k|\omega|-j\omega t_d}$ (where k is the loss constant and t_d the channel delay), an expression for the *jitter amplification factor* A_{RJ} in the case of random jitter can be derived, as displayed in Equation 26 in [138]:

$$A_{RJ} = \sqrt{\frac{5}{\ln(10)\left|L\left(f_{Nyq}\right)\right|}\sinh\left(\frac{\ln(10)}{10}\left|L\left(f_{Nyq}\right)\right|\right) + \frac{1}{2}}, \tag{A.38}$$

where $L(f_{Nyq})$ is the channel loss at the Nyquist frequency.

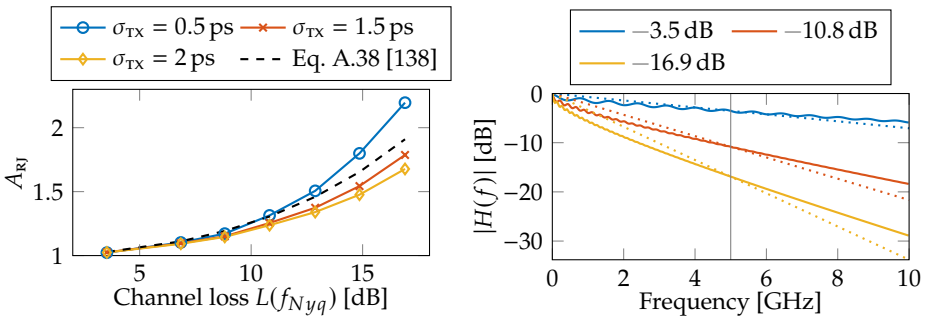
Jitter Measurement	Definition	Description
Type A	$t_{s,e} - t_{ck,ideal}$	Time difference between the CDR's clock and an ideal reference clock.
Type B	$t_{s,e} - t_{s,e\text{-}id}$	Time difference between the CDR's clock and that of a quasi-ideal linear CDR.
Type C	$t_{s,e} - t_{0\text{-}X}$	Time difference between the CDR's clock and the zero-crossings at the receiver.
Type D	$t_{0\text{-}X} - t_{TX}$	Time difference between $v(t)$'s and the transmitter's zero-crossings.
Type E	$t_{0\text{-}X} - t_{ck,ideal}$	Time difference between the zero-crossings of the received waveform and an ideal reference clock.

Table A.1: Definitions of the various options for defining jitter in presence of ISI.

Figure A.19a reports jitter amplification factors A_{RJ} as a function of channel loss for different values of transmitter's clock random jitter alongside a plot of Equation A.38 as a term of comparison; for this purpose, type D jitter was evaluated and its standard deviation compared to that of the jitter introduced at the transmitter. Figure A.19b compares the channels used in the simulator with ISI to the corresponding approximate loss models assumed in [138] to derive Equation A.38. Notice the reduced accuracy of the approximation with respect to the simulated channels as the loss increases, which is likely to be responsible for the discrepancy between simulations and Equation A.38 at high attenuation values (above 10 dB).

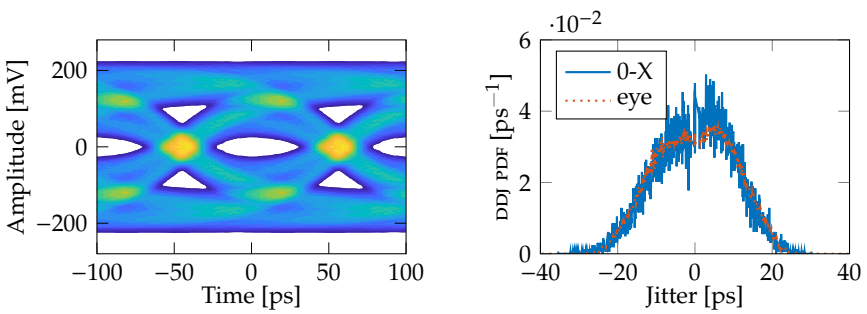
Another test was performed by comparing the data-dependent jitter extracted from the signal crossings at the receiver (i.e. type E jitter of Table A.1) and that obtained by cutting the eye diagram at the transitions, in order to verify that the two methods are equivalent. The result is shown in Figure A.20 for a channel losing approximately 13 dB at 5 GHz and shows a good correspondence between the two approaches. Notice that extracting the data-dependent jitter from the eye diagram comes with little extra effort due to the statistical method employed and the fact that the eye diagram is typically computed anyway to assess reception quality, whereas the whole received analog waveform has to be scanned to search for the signal transitions, moreover requiring interpolation to determine their precise timing despite the analog samples being spaced by δt.

The operation of the simulator can also be shown when operating in a realistic application scenario, as that of a channel losing approximately 13 dB at a Nyquist frequency of 5 GHz with an FFE having weights $w = (-0.1, 0.85, -0.05)$ applied at the transmitter. The ideal transmitted sequence and the resulting waveform at the receiver's slicers (i.e. after the channel and the AFE) are shown in Figure A.21a; notice that alignment between the two had to be performed manually, because the channel adds a propagation delay to the input signal, as mentioned above. Figure A.21b shows histograms of the various jitter components, indicating that the gaussian absolute jitter having 1 ps standard deviation introduced at the transmitter, although unbounded, probably has a smaller impact than the much wider data-dependent jitter induced by the channel (computed through the eye diagram); the type A jitter at the output of the CDR operating at full rate with a 32-step PI and



(a) Jitter amplification (b) Channel transfer function

Figure A.19: Jitter amplification compared to the results in [138]: a) A_{RJ} (solid lines) to a clock-like bit sequence as a function of channel loss for various jitter variances of the transmitter's clock, and comparison with Equation A.38 (dashed line, Equation 26 in [138]); and b) three of the channels transfer functions employed in the simulator with ISI compared to the corresponding approximate loss models (intersecting at $L(f_{Nyq}) = 5\,\text{GHz}$, vertical gray line) assumed in [138].



(a) (b)

Figure A.20: (a) Eye diagram after a channel losing approximately 13 dB at 5 GHz and (b) comparison between the data-dependent jitter distributions (centred on the mean transition time) computed through the eye diagram obtained with the statistical simulator of Chapter 2 ("eye") and as type E jitter ("0-X").

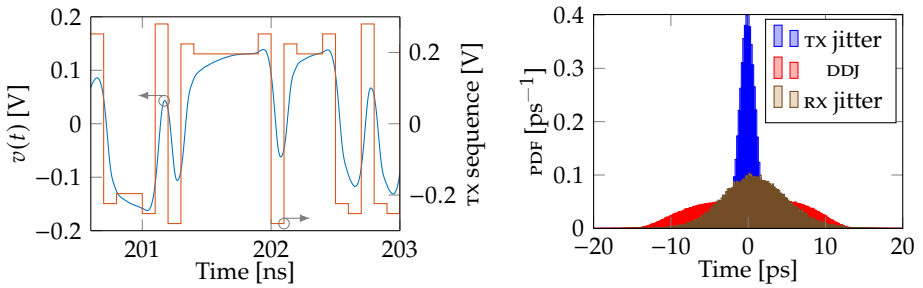


Figure A.21: (a) Analog received waveform $v(t)$ and transmitter's amplitude sequence after an FFE applying $w = (-0.1,\ 0.85,\ -0.05)$ (the two were aligned manually); (b) histograms of jitter added at the transmitter, due to ISI only as type E and at the output of the CDR under test as type A ($N_{\text{PI}} = 32$, $N_{\text{div}} = 1$, $N_{\text{des}} = 1$).

$N_{\text{div}} = 1$ is also reported to assess its performance.

A more direct comparison between the simulators without and with ISI is shown in Figure A.22, where the CDR's type A jitter as a function of the period jitter of the receiver's oscillator is reported for the case without data-dependent jitter of Figure A.6 and for a few channels with different losses. As can be seen, the impact of ISI-induced jitter is minimal for the low-loss channel, whose resulting jitter is similar to that obtained with the simulator without ISI, whereas channels featuring higher losses introduce relevant data-dependent jitter that the CDR cannot track, hence increasing the jitter at its output.

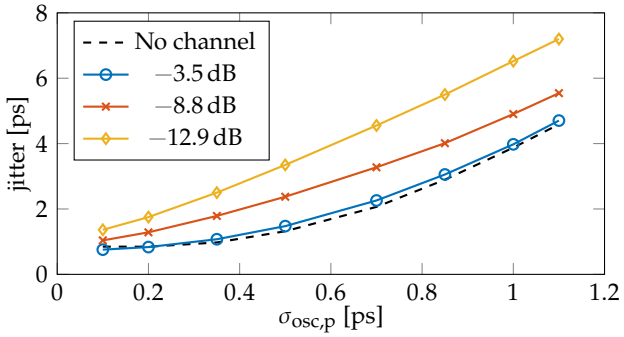


Figure A.22: Simulated type A jitter for a CDR operating at 10 Gb/s on the serial data considering a free-running oscillator as the transmitter clock source; $N_{\text{PI}} = 64$, $N_{\text{div}} = 4$. The dashed black line is the case without ISI of Figure A.6, whereas the other curves (blue, red and orange) were obtained for channel losses of −3.5 dB, −8.8 dB and −12.9 dB, respectively, all referred to the Nyquist frequency; no FFE was applied.

A.4 An Overview on Power Supply-Induced Jitter

As was briefly mentioned throughout this thesis, one relevant source of jitter is the noise present on the supplies due to various interferers such as the switching activity of the circuit itself, which alters the operating point, output swing and transition threshold of the various blocks and thus the timing of their operations, effectively giving rise to jitter at their output: This is called *power supply-induced jitter (PSIJ)* [33,63]. As such, it is ubiquitous in electronic circuits simply because it is impossible to perfectly decouple the supplies from all possible noise sources, and must be coped with by analog/mixed-signal designers and layouters to prevent it from hampering a circuit's performance.

The circuit's switching activity becomes a source of noise when the transient currents drawn from/sunk to the supplies pass through the *power delivery network (PDN)*, which comprises on- and off-chip interconnections, voltage regulators and package parasitics, thus giving rise to voltage noise [33, 63]. Such networks are typically modelled as combinations of resistors, inductors and capacitors, resonating at few to hundreds of MHz and generating voltage noise amplitudes of tens of mV [33, 63].

Given such causes of PSIJ, it is clear that it is highly dependent on the activity pattern of the switching circuit (e.g. random for blocks dealing with data, whereas concentrated at a fixed frequency in a clock buffering chain), hence being a form of data-dependent jitter, and that it is exacerbated by high bit rates, fast rise/fall times and increased I/O pin count [63].

A vast literature is devoted to the topic of PSIJ modelling with the aim of predicting it and then determine how to reduce its impact. One of the simplest approaches to PSIJ estimation is a delay-based method that simply considers how the propagation delay t_d of a certain (typically simple and small, such as an inverter) circuit is affected by a voltage supply differing from the nominal V_DD by ΔV_DD: For small enough variations, the *PSIJ sensitivity* (or *alpha factor*) α can be employed to express the relation

$$\frac{\Delta t_\mathrm{d}}{t_\mathrm{d}} = \alpha \frac{\Delta V_\mathrm{DD}}{V_\mathrm{DD}}, \tag{A.39}$$

which can then be used to derive analytical expressions for α in terms of the circuit's parameters, such as nominal supply and transistor's threshold voltage [33, 63]. As an example, assuming that the square-law model holds for a CMOS inverter during an output falling edge transition, the simple formula

$$\alpha = -\frac{V_\mathrm{DD} + V_\mathrm{th}}{V_\mathrm{DD} - V_\mathrm{th}} \tag{A.40}$$

can be obtained [33]. Of course, for deep sub-µm technologies such a simplified expression is not strictly valid and the transistor's non-linear behaviour increases inaccuracy on α, especially at high frequencies and large noise amplitudes, although it can still provide useful insight on PSIJ [33, 63].

Changing perspective, frequency-domain analyses can provide useful results to abstract from the implementation of single circuits and focus instead on the whole system or at least on blocks residing under the different supply domains commonly present in integrated circuits [63, 154]. First, the supply noise spectrum is derived as

$$V(f) = Z_\mathrm{PDN}(f) I_\mathrm{noise}(f), \tag{A.41}$$

where $Z_{\text{PDN}}(f)$ is the PDN's impedance and $I_{\text{noise}}(f)$ is the spectrum of the current drawn by the system, both of them determined through simulation. Then, the *jitter sensitivity profile* $S(f)$ is extracted by superposing a low-amplitude sinusoid to the voltage supply and measuring the resulting timing error with respect to the noiseless case; by repeating such a procedure for all the noise frequencies of interest, a complete profile is obtained [63], or it can be derived analytically for open-loop subcircuit paths using the DC delay sensitivity, computed from two different delay values estimated at different voltages [154]. Eventually, the jitter spectrum is computed by combining supply noise and system's sensitivity as

$$J(f) = V(f)S(f), \tag{A.42}$$

from which peak-to-peak jitter can be calculated and inverse-Fourier transform of $J(f)$ can be performed to obtain the time-domain jitter sequence corresponding to the circuit activity that resulted in $I_{\text{noise}}(f)$ [63].

Many other modelling approaches for PSIJ have been proposed in the past, from semi-analytical techniques that use simplified MOSFET equations with simulated parameters to predict jitter at the end of inverter chains, to methods based on estimating the jitter PDF from linear model of transistors; from piecewise-linear/non-linear models to approximate a MOSFET's I-V curve in order to determine how the transition's midpoint is affected by supply noise, to techniques based on *input/output buffer information specification* (IBIS) macromodels of the given circuits [63].

An interesting approach used to include PSIJ in behavioural models is presented in [141] for the case of a CDR, where the effect of supply noise on the phase detector's sampling time is considered by exploiting the concept of *impulse sensitivity function* (ISF) $\Gamma(\tau)$: It expresses the variation in sampling time as a function of the time difference τ between the clock's sampling edge and a noise impulse applied on the power supply, so that the sampling time offset at the n-th clock edge is

$$\Delta\phi_n = \int_{-\infty}^{+\infty} V_{\text{noise}}(\tau - t_n)\Gamma(\tau)\mathrm{d}\tau, \tag{A.43}$$

where V_{noise} is generated by a generic noise source. The complete profile of $\Gamma(\tau)$ is extracted from circuit simulations by sweeping the time τ of the noise impulse and measuring the resulting time offset; then, Equation A.43 can be included in the behavioural models to account for PSIJ, provided that supply noise is also comprised in the simulation framework [141].

Bibliography

[1] B. Razavi, "Historical Trends in Wireline Communications: 60x Improvement in Speed in 20 Years," *IEEE Solid-State Circuits Mag.*, vol. 7, no. 4, pp. 42–46, Dec. 2015.

[2] H. Tamura, "Looking to the Future: Projected Requirements for Wireline Communications Technology," *IEEE Solid-State Circuits Mag.*, vol. 7, no. 4, pp. 53–62, 2015.

[3] X. Zheng, C. Zhang, F. Lv, F. Zhao, S. Yuan, S. Yue, Z. Wang, F. Li, Z. Wang, and H. Jiang, "A 40-Gb/s Quarter-Rate SerDes Transmitter and Receiver Chipset in 65-nm CMOS," *IEEE J. Solid-State Circuits*, vol. 52, no. 11, pp. 2963–2978, Nov. 2017.

[4] S. Palermo, S. Hoyos, S. Cai, S. Kiran, and Y. Zhu, "Analog-to-Digital Converter-Based Serial Links: An Overview," *IEEE Solid-State Circuits Mag.*, vol. 10, no. 3, pp. 35–47, Aug. 2018.

[5] B. Dehlaghi, N. Wary, and T. C. Carusone, "Ultra-Short-Reach Interconnects for Die-to-Die Links: Global Bandwidth Demands in Microcosm," *IEEE Solid-State Circuits Mag.*, vol. 11, no. 2, pp. 42–53, Jun. 2019.

[6] S. Kiran, S. Cai, Y. Zhu, S. Hoyos, and S. Palermo, "Digital Equalization With ADC-Based Receivers: Two Important Roles Played by Digital Signal Processingin Designing Analog-to-Digital-Converter-Based Wireline Communication Receivers," *IEEE Microwave Mag.*, vol. 20, no. 5, pp. 62–79, May 2019.

[7] D. C. Daly, L. C. Fujino, and K. C. Smith, "Through the Looking Glass-2020 Edition: Trends in Solid-State Circuits From ISSCC," *IEEE Solid-State Circuits Mag.*, vol. 12, no. 1, pp. 8–24, Jan. 2020.

[8] T. C. Carusone, "Introduction to Digital I/O: Constraining I/O Power Consumption in High-Performance Systems," *IEEE Solid-State Circuits Mag.*, vol. 7, no. 4, pp. 14–22, Fall 2015.

[9] J. Gao, H. Cheng, H.-C. Wu, G. Liu, E. Lau, L. Yuan, and C. Krause, "Thunderbolt Interconnect—Optical and Copper," *Journal of Lightwave Technology*, vol. 35, no. 15, pp. 3125–3129, Aug. 2017.

[10] K. Hu, R. Bai, T. Jiang, C. Ma, A. Ragab, S. Palermo, and P. Y. Chiang, "0.16-0.25 pJ/bit, 8 Gb/s Near-Threshold Serial Link Receiver With Super-Harmonic Injection-Locking," *IEEE J. Solid-State Circuits*, vol. 47, no. 8, pp. 1842–1853, Aug. 2012.

[11] A. Tajalli, A. Hormati, and A. Shokrollahi, "Chord Signaling for High-Speed Data Movement: Employing Advanced Communication and Circuit Techniques to Augment Data-Transfer Bandwidth," *IEEE Solid-State Circuits Mag.*, vol. 11, no. 2, pp. 78–85, Jun. 2019.

[12] K. Chang, G. Zhang, and C. Borrelli, "Evolution of Wireline Transceiver Standards: Various, Most-Used Standards for the Bandwidth Demand," *IEEE Solid-State Circuits Mag.*, vol. 7, no. 4, pp. 47–52, Dec. 2015.

[13] B. Casper, "Clocking Wireline Systems: An Overview of Wireline Design Techniques," *IEEE Solid-State Circuits Mag.*, vol. 7, no. 4, pp. 32–41, 2015.

[14] W. Zeng, M. A. S. Khalid, and S. Chowdhury, "In-Vehicle Networks Outlook: Achievements and Challenges," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 3, pp. 1552–1571, 2016.

[15] E. Depaoli, H. Zhang, M. Mazzini, W. Audoglio, A. A. Rossi, G. Albasini, M. Pozzoni, S. Erba, E. Temporiti, and A. Mazzanti, "A 64 Gb/s Low-Power Transceiver for Short-Reach PAM-4 Electrical Links in 28-Nm FDSOI CMOS," *IEEE J. Solid-State Circuits*, vol. 54, no. 1, pp. 6–17, 2019.

[16] R. Yousry *et al.*, "11.1 A 1.7pJ/b 112Gb/s XSR Transceiver for Intra-Package Communication in 7nm FinFET Technology," in *2021 IEEE Int. Solid State Circuits Conf. - Dig. of Tech. Papers*, 2021, pp. 180–182.

[17] J. M. Wilson, W. J. Turner, and J. W. Poulton, "Ground-Referenced Single-Ended Signaling: Applications for High-Density, Short-Haul Communication Systems," *IEEE Solid-State Circuits Mag.*, vol. 11, no. 2, pp. 54–68, Jun. 2019.

[18] C. Kim, H.-W. Lee, and J. Song, "Memory Interfaces: Past, Present, and Future," *IEEE Solid-State Circuits Mag.*, vol. 8, no. 2, pp. 23–34, Jun. 2016.

[19] A. Karkar, T. Mak, K.-F. Tong, and A. Yakovlev, "A Survey of Emerging Interconnects for On-Chip Efficient Multicast and Broadcast in Many-Cores," *IEEE Circuits Syst. Mag.*, vol. 16, no. 1, pp. 58–72, Feb. 2016.

[20] M. Hekmat, S. Song, N. Jaffari, S. Sankaranarayanan, C. Huang, M. Han, G. Malhotra, J. Kamali, A. Amirkhany, and W. Xiong, "23.3 A 6Gb/s 3-tap FFE transmitter and 5-tap DFE receiver in 65nm/0.18μm CMOS for next-generation 8K displays," in *2016 IEEE Int. Solid-State Circuits Conf. (ISSCC)*, Jan. 2016, pp. 402–403.

[21] J. Park, J.-H. Chae, Y.-U. Jeong, J.-W. Lee, and S. Kim, "A 2.1-Gb/s 12-Channel Transmitter With Phase Emphasis Embedded Serializer for 55-in UHD Intra-Panel Interface," *IEEE J. Solid-State Circuits*, vol. 53, no. 10, pp. 2878–2888, Oct. 2018.

[22] Y. Komatsu, A. Shinmyo, S. Kato, M. Funabashi, K. Hatooka, M. Fujita, K. Tanaka, A. Yamasaki, and K. Fukuda, "A 0.25-27-Gb/s PAM4/NRZ Transceiver With Adaptive Power CDR and Jitter Analysis," *IEEE J. Solid-State Circuits*, pp. 1–10, 2019.

[23] J. Lee, K. Lee, H. Kim, B. Kim, K. Park, and D.-K. Jeong, "A 0.1-pJ/b/dB 1.62-to-10.8-Gb/s Video Interface Receiver With Jointly Adaptive CTLE and DFE Using Biased Data-Level Reference," *IEEE J. Solid-State Circuits*, vol. 55, no. 8, pp. 2186–2195, Aug. 2020.

[24] A. Bathini, M. Jayasimha, and D. Nagaraj, "Signal Integrity Challenges and Solutions for USB4 and TBT3 Protocols," in *2020 IEEE Electrical Design of Adv. Packag. and Syst. (EDAPS)*, 2020, pp. 1–3.

[25] F. Gerfers, "Basics of DAC-based Wireline Transmitters," 2021 IEEE Int. Solid State Circuits Conf., Short Course, San Francisco, CA, USA, Feb. 2021.

[26] M. Daily, S. Medasani, R. Behringer, and M. Trivedi, "Self-Driving Cars," *Computer*, vol. 50, no. 12, pp. 18–23, Dec. 2017.

[27] M. Traub, A. Maier, and K. L. Barbehon, "Future Automotive Architecture and the Impact of IT Trends," *IEEE Software*, vol. 34, no. 3, pp. 27–32, May 2017.

[28] J. P. Trovao, "Trends in Automotive Electronics [Automotive Electronics]," *IEEE Vehicular Technol. Mag.*, vol. 14, no. 4, pp. 100–109, Dec. 2019.

[29] K. Matheus, "Evolution of Ethernet-Based Automotive Networks: Faster and Cheaper," London, 2018.

[30] L. Lo Bello, R. Mariani, S. Mubeen, and S. Saponara, "Recent Advances and Trends in On-Board Embedded and Networked Automotive Systems," *IEEE Trans. on Industrial Informatics*, vol. 15, no. 2, pp. 1038–1051, Feb. 2019.

[31] J. F. Bulzacchelli, "Equalization for Electrical Links: Current Design Techniques and Future Directions," *IEEE Solid-State Circuits Mag.*, vol. 7, no. 4, pp. 23–31, Fall 2015.

[32] B. Casper and F. O'Mahony, "Clocking Analysis, Implementation and Measurement Techniques for High-Speed Data Links—A Tutorial," *IEEE Trans. Circuits Syst. I*, vol. 56, no. 1, pp. 17–39, Jan. 2009.

[33] X. Mo, J. Wu, N. Wary, and T. C. Carusone, "Design Methodologies for Low-Jitter CMOS Clock Distribution," *IEEE Open J. of the Solid-State Circuits Society*, vol. 1, pp. 94–103, Oct. 2021.

[34] A. Amirkhany, "Basics of Clock and Data Recovery Circuits: Exploring High-Speed Serial Links," *IEEE Solid-State Circuits Mag.*, vol. 12, no. 1, pp. 25–38, Jan. 2020.

[35] P. Palestri, A. Elnaqib, D. Menin, K. Shyti, F. Brandonisio, A. Bandiziol, D. Rossi, and R. Nonis, "Analytical Modeling of Jitter in Bang-Bang CDR Circuits Featuring Phase Interpolation," *IEEE Trans. VLSI Syst.*, vol. 29, no. 7, pp. 1392–1401, 2021.

[36] J. F. Bulzacchelli *et al.*, "A 28-Gb/s 4-Tap FFE/15-Tap DFE Serial Link Transceiver in 32-nm SOI CMOS Technology," *IEEE J. Solid-State Circuits*, vol. 47, no. 12, pp. 3232–3248, Dec. 2012.

[37] M. Abu Khater, "High-Speed Printed Circuit Boards: A Tutorial," *IEEE Circuits Syst. Mag.*, vol. 20, no. 3, pp. 34–45, Aug. 2020.

[38] A. von Hippel, *Dielectrics and Waves*. Artech House, 1995, ch. Complex Permittivity and Permeability, pp. 3–6.

[39] K. L. Chan *et al.*, "A 32.75-Gb/s Voltage-Mode Transmitter With Three-Tap FFE in 16-nm CMOS," *IEEE J. Solid-State Circuits*, vol. 52, no. 10, pp. 2663–2678, Oct. 2017.

[40] L. Zhang and T. Kwasniewski, "Optimal Equalization for Reducing the Impact of Channel Group Delay Distortion on High-Speed Backplane Data Transmission," *AEU - International Journal of Electronics and Communications*, vol. 64, no. 7, pp. 671–681, 2009.

[41] J. Im *et al.*, "A 40-to-56 Gb/s PAM-4 Receiver With Ten-Tap Direct Decision-Feedback Equalization in 16-nm FinFET," *IEEE J. Solid-State Circuits*, vol. 52, no. 12, pp. 3486–3502, Dec. 2017.

[42] J. W. M. Bergmans, *Digital Baseband Transmission and Recording*. Springer, 1996, ch. Adaptive Reception, pp. 373–450.

[43] J. G. Proakis and M. Salehi, *Digital communications*, 5th ed. Boston: McGraw-Hill, 2008.

[44] N. Da Dalt and A. Sheikholeslami, *Understanding Jitter and Phase Noise: A Circuits and Systems Perspective*, 1st ed. Cambridge University Press, Feb. 2018.

[45] G. Balamurugan, B. Casper, J. Jaussi, M. Mansuri, F. O'Mahony, and J. Kennedy, "Modeling and Analysis of High-Speed I/O Links," *IEEE Trans. Adv. Packag.*, vol. 32, no. 2, pp. 237–247, May 2009.

[46] B. Razavi, "Design Techniques for High-Speed Wireline Transmitters," *IEEE Open J. of the Solid-State Circuits Society*, vol. 1, pp. 53–66, 2021.

[47] B. Widrow, J. M. McCool, M. G. Larimore, and C. R. Johnson, "Stationary and nonstationary learning characteristics of the LMS adaptive filter," *Proc. IEEE*, vol. 64, no. 8, pp. 1151–1162, 1976.

[48] A. Sheikholeslami, "AC Coupling and Baseline Wander [Circuit Intuitions]," *IEEE Solid-State Circuits Mag.*, vol. 13, no. 3, pp. 13–16, Aug. 2021.

[49] G. Kim *et al.*, "A 161-mW 56-Gb/s ADC-Based Discrete Multitone Wireline Receiver Data-Path in 14-nm FinFET," *IEEE J. Solid-State Circuits*, vol. 55, no. 1, pp. 38–48, Jan. 2020.

[50] B. Vatankhahghadim, N. Wary, J. Bailey, and A. Chan Carusone, "A Study of Discrete Multitone Modulation for Wireline Links Beyond 100 Gb/s," *IEEE Open J. of Circuits and Syst.*, vol. 2, pp. 78–90, 2021.

[51] A. Tajalli *et al.*, "A 1.02-pJ/b 20.83-Gb/s/Wire USR Transceiver Using CNRZ-5 in 16-nm FinFET," *IEEE J. Solid-State Circuits*, vol. 55, no. 4, pp. 1108–1123, Apr. 2020.

[52] N. J. Endo, "Wireless Communication In and Around the Car: Status and Outlook. ES3: High-Speed Communications on 4 Wheels: What's in Your next Car?" in *2013 IEEE Int. Solid-State Circuits Conf. Dig. of Tech. Papers*, Feb. 2013, pp. 515–515.

[53] G. W. den Besten, "30.1 Single-Pair Automotive PHY Solutions from 10Mb/s to 10Gb/s and Beyond," in *2019 IEEE Int. Solid State Circuits Conf. - Dig. of Tech. Papers*, 2019, pp. 474–476.

[54] A. Bandiziol, "Modeling, Design and Characterization of a 10Gbps Serial Link," Ph.D. dissertation, Università degli Studi di Udine, 2017.

[55] A. Bandiziol, W. Grollitsch, G. Steffan, R. Nonis, and P. Palestri, "Design and Characterization of a 9.2-Gb/s Transceiver for Automotive Microcontroller Applications With 8-Taps FFE and 1-Tap Unrolled/4-Taps DFE," *IEEE Trans. Circuits Syst. II*, vol. 65, no. 10, pp. 1305–1309, Oct. 2018.

[56] J. Ren and K. S. Oh, "Multiple Edge Responses for Fast and Accurate System Simulations," *IEEE Trans. Adv. Packag.*, vol. 31, no. 4, pp. 741–748, Nov. 2008.

[57] V. Stojanović and M. Horowitz, "Modeling and analysis of high-speed links," in *Proc. of the IEEE 2003 Custom Integrated Circuits Conf., 2003.* IEEE, 2003, pp. 589–594.

[58] A. Sanders, "Statistical Simulation of Physical Transmission Media," *IEEE Trans. Adv. Packag.*, vol. 32, no. 2, pp. 260–267, May 2009.

[59] A. Cristofoli, P. Palestri, N. D. Dalt, and L. Selmi, "Efficient Statistical Simulation of Intersymbol Interference and Jitter in High-Speed Serial Interfaces," *IEEE Trans. Compon. Packag. Manuf. Technol.*, vol. 4, no. 3, pp. 480–489, Mar. 2014.

[60] D. Menin, A. De Prà, A. Bandiziol, W. Grollitsch, R. Nonis, and P. Palestri, "A Simple Simulation Approach for the Estimation of Convergence and Performance of Fully-Adaptive Equalization in High-Speed Serial Interfaces," *IEEE Trans. Compon. Packag. Manuf. Technol.*, vol. 9, no. 10, pp. 2079–2086, Oct. 2019.

[61] D. Menin *et al.*, "A Simple Modelling Tool for Fast Combined Simulation of Interconnections, Inter-Symbol Interference and Equalization in High-Speed Serial Interfaces for Chip-to-Chip Communications," *Advances in Science, Technol. and Eng. Systems J.*, vol. 5, no. 2, pp. 527–536, 2020.

[62] G. Signorini, C. Siviero, M. Telescu, and I. S. Stievano, "Present and Future of I/O-Buffer Behavioral Macromodels," *IEEE Electromagn. Compat.*, vol. 5, no. 3, pp. 79–85, 2016.

[63] J. N. Tripathi, V. K. Sharma, and H. Shrimali, "A Review on Power Supply Induced Jitter," *IEEE Trans. Compon. Packag. Manuf. Technol.*, vol. 9, no. 3, pp. 511–524, Mar. 2019.

[64] M. Dazzi, "Modeling and Design of a Serial Interface for Chip-to-Chip Communication at 1 Gbps," Master's thesis, Università degli Studi di Udine, 2016.

[65] M. Dazzi, P. Palestri, D. Rossi, A. Bandiziol, I. Loi, D. Bellasi, and L. Benini, "Sub-mW Multi-Gbps Chip-to-Chip Communication Links for Ultra-Low Power IoT End-Nodes," in *2018 IEEE Int. Symp. on Circuits and Syst. (ISCAS)*, May 2018, pp. 1–5.

[66] E. Denlinger, "Losses of Microstrip Lines," *IEEE Trans. Microw. Theory Techn.*, vol. 28, no. 6, pp. 513–522, Jun. 1980.

[67] W. Getsinger, "Microstrip Dispersion Model," *IEEE Trans. Microw. Theory Techn.*, vol. 21, no. 1, pp. 34–39, Jan. 1973.

[68] E. Hammerstad and O. Jensen, "Accurate Models for Microstrip Computer-Aided Design," in *MTT-S Int. Microwave Symp. Dig.*, vol. 80. MTT006, 1980, pp. 407–409.

[69] M. Kirschning and R. Jansen, "Accurate Model for Effective Dielectric Constant of Microstrip with Validity up to Millimetre-Wave Frequencies," *Electron. Lett.*, vol. 18, no. 6, p. 272, 1982.

[70] H. Wheeler, "Transmission-Line Properties of a Strip on a Dielectric Sheet on a Plane," *IEEE Trans. Microw. Theory Techn.*, vol. 25, no. 8, pp. 631–647, Aug. 1977.

[71] M. Kirschning and R. Jansen, "Arguments and an Accurate Model for the Power-Current Formulation of Microstrip Characteristic Impedance," *Archiv der elektrischen Übertragung: AEÜ*, vol. 37, pp. 108–112, mar 1983.

[72] M. Kirschning and R. Jansen, "Accurate Wide-Range Design Equations for the Frequency-Dependent Characteristic of Parallel Coupled Microstrip Lines," *IEEE Trans. Microw. Theory Techn.*, vol. 32, no. 1, pp. 83–90, Jan. 1984.

[73] T. Bernardi, "Modelli per Linee di Trasmissione Accoppiate in Sistemi di Comunicazione Chip-to-Chip," BSc, Università degli Studi di Udine, Udine, 2018.

[74] T. Brazil, "Causal-convolution – a New Method for the Transient Analysis of Linear Systems at Microwave Frequencies," *IEEE Trans. Microw. Theory Techn.*, vol. 43, no. 2, pp. 315–323, Feb. 1995.

[75] Alexander, J.D.H., "Clock Recovery from Random Binary Signals," *Electron. Lett.*, vol. 11, no. 22, pp. 541–542, Oct. 1975.

[76] D. Menin, A. Bandiziol, W. Grollitsch, R. Nonis, and P. Palestri, "Design and Simulation of a 12 Gb/s Transceiver with 8-Tap FFE, Offset-Compensated Samplers and Fully-Adaptive 1-Tap Speculative/3-Tap DFE and Sampling Phase for MIPI A-PHY Applications," *IEEE Trans. Circuits Syst. II*, vol. 67, no. 8, pp. 1369–1373, 2020.

[77] A. Sanders, M. Resso, and J. D'Ambrosia, "Channel Compliance Testing Utilizing Novel Statistical Eye Methodology," in *Euro DesignCon 2004*, 2004.

[78] V. Stojanović *et al.*, "Autonomous Dual-Mode (PAM2/4) Serial Link Transceiver with Adaptive Equalization and Data Recovery," *IEEE J. Solid-State Circuits*, vol. 40, no. 4, pp. 1012–1026, Apr. 2005.

[79] V. Balan, O. Oluwole, G. Kodani, C. Zhong, R. Dadi, A. Amin, A. Ragab, and M.-J. E. Lee, "A 15–22 Gbps Serial Link in 28 nm CMOS With Direct DFE," *IEEE J. Solid-State Circuits*, vol. 49, no. 12, pp. 3104–3115, 2014.

[80] N. Neurohr, M. Schoebinger, E. Prete, and A. Sanders, "Adaptive Decision-Feedback Equalization for Band-Limited High-Speed Serial Links," in *2005 IEEE Int. Symp. of Circuits and Syst. (ISCAS)*, vol. 2, 2005, pp. 924–927.

[81] R. W. Lucky, "Techniques for Adaptive Equalization of Digital Communication Systems," *The Bell System Technical J.*, vol. 45, no. 2, pp. 255–286, Feb 1966.

[82] B. Widrow and J. M. J. Hoff, "Adaptive Switching Circuits," *IRE Wescon Conv. Rec.*, vol. 4, pp. 96–104, Aug. 1960.

[83] R. L. Burden and J. D. Faires, *Numerical Analysis*, 9th ed. Pacific Grove, Calif.: Brooks/Cole Cengage Learning, 2011.

[84] R. W. Lucky, "Automatic Equalization for Digital Communication," *The Bell System Technical J.*, vol. 44, no. 4, pp. 547–588, April 1965.

[85] S. Dasgupta, C.R. Johnson, and A.M. Baksho, "Sign-Sign LMS Convergence with Independent Stochastic Inputs," *IEEE Trans. Inf. Theory*, vol. 36, no. 1, pp. 197–201, Jan. 1990.

[86] H.-J. Chi, J.-S. Lee, S.-H. Jeon, S.-J. Bae, Y.-S. Sohn, J.-Y. Sim, and H.-J. Park, "A Single-Loop SS-LMS Algorithm With Single-Ended Integrating DFE Receiver for Multi-Drop DRAM Interface," *IEEE J. Solid-State Circuits*, vol. 46, no. 9, pp. 2053–2063, Sep. 2011.

[87] G. R. Gangasani *et al.*, "A 32 Gb/s Backplane Transceiver With On-Chip AC-Coupling and Low Latency CDR in 32 nm SOI CMOS Technology," *IEEE J. Solid-State Circuits*, vol. 49, no. 11, pp. 2474–2489, Nov. 2014.

[88] J. Savoj *et al.*, "Wideband Flexible-Reach Techniques for a 0.5-16.3Gb/s Fully-Adaptive Transceiver in 20nm CMOS," in *Proc. of the IEEE 2014 Custom Integrated Circuits Conf.* IEEE, Sep. 2014, pp. 1–4.

[89] J. Han, Y. Lu, N. Sutardja, K. Jung, and E. Alon, "A 60Gb/s 173mW Receiver Frontend in 65nm CMOS Technology," in *2015 Symp. on VLSI Circuits (VLSI Circuits)*, June 2015, pp. C230–C231.

[90] T. Shibasaki *et al.*, "A 56Gb/s NRZ-Electrical 247mW/Lane Serial-Link Transceiver in 28nm CMOS," in *2016 IEEE Int. Solid-State Circuits Conf. (ISSCC)*. IEEE, Jan. 2016, pp. 64–65.

[91] J. Savoj *et al.*, "A Low-Power 0.5–6.6 Gb/s Wireline Transceiver Embedded in Low-Cost 28 nm FPGAs," *IEEE J. Solid-State Circuits*, vol. 48, no. 11, pp. 2582–2594, Nov. 2013.

[92] P. Aziz and A. Malipatil, "Adaptation Algorithms for a Class of Continuous Time Analog Equalizers with Application to Serial Links," in *2011 IEEE Int. Symp. of Circuits and Syst. (ISCAS)*. IEEE, 2011, pp. 1383–1386.

[93] H. Kimura *et al.*, "A 28 Gb/s 560 mW Multi-Standard SerDes With Single-Stage Analog Front-End and 14-Tap Decision Feedback Equalizer in 28 nm CMOS," *IEEE J. Solid-State Circuits*, vol. 49, no. 12, pp. 3091–3103, 2014.

[94] H. Higashi *et al.*, "A 5-6.4-Gb/s 12-Channel Transceiver with Pre-Emphasis and Equalization," *IEEE J. Solid-State Circuits*, vol. 40, no. 4, pp. 978–985, Apr 2005.

[95] W.-S. Kim, C.-K. Seong, and W.-Y. Choi, "A 5.4-Gbit/s Adaptive Continuous-Time Linear Equalizer Using Asynchronous Undersampling Histograms," *IEEE Trans. Circuits Syst. II*, vol. 59, no. 9, pp. 553–557, 2012.

[96] Y.-M. Ying and S.-I. Liu, "A 20Gb/s Digitally Adaptive Equalizer/DFE with Blind Sampling," in *2011 IEEE Int. Solid-State Circuits Conf.* IEEE, 2011, pp. 444–446.

[97] Y.-F. Lin, C.-C. Huang, J.-Y. M. Lee, C.-T. Chang, and S.-I. Liu, "A 5-20 Gb/s Power Scalable Adaptive Linear Equalizer Using Edge Counting," in *2014 IEEE Asian Solid-State Circuits Conf. (A-SSCC)*, 2014, pp. 273–276.

[98] D. Yoo, M. Bagherbeik, W. Rahman, A. Sheikholeslami, H. Tamura, and T. Shibasaki, "A 36-Gb/s Adaptive Baud-Rate CDR With CTLE and 1-Tap DFE in 28-nm CMOS," *IEEE Solid-State Circuits Lett.*, vol. 2, no. 11, pp. 252–255, 2019.

[99] H. Won, J.-Y. Lee, T. Yoon, K. Han, S. Lee, J. Park, and H.-M. Bae, "A 28-Gb/s Receiver With Self-contained Adaptive Equalization and Sampling Point Control Using Stochastic Sigma-Tracking Eye-Opening Monitor," *IEEE Trans. Circuits Syst. I*, vol. 64, no. 3, pp. 664–674, 2017.

[100] G. E. Zhang and M. M. Green, "A 10 Gb/s BiCMOS Adaptive Cable Equalizer," *IEEE J. Solid-State Circuits*, vol. 40, no. 11, pp. 2132–2140, 2005.

[101] Y.-H. Tu, K.-H. Cheng, M.-J. Lee, and J.-C. Liu, "A Power-Saving Adaptive Equalizer With a Digital-Controlled Self-Slope Detection," *IEEE Trans. Circuits Syst. I*, vol. 65, no. 7, pp. 2097–2108, 2018.

[102] H. Liu, I. Mohammed, Y. Fan, M. Morgan, and J. Liu, "An HDMI Cable Equalizer With Self-Generated Energy Ratio Adaptation Scheme," *IEEE Trans. Circuits Syst. II*, vol. 56, no. 7, pp. 595–599, 2009.

[103] H.-Y. Joo and L.-S. Kim, "A Data-Pattern-Tolerant Adaptive Equalizer Using the Spectrum Balancing Method," *IEEE Trans. Circuits Syst. II*, vol. 57, no. 3, pp. 228–232, 2010.

[104] B. Dehlaghi *et al.*, "A 1.41-pJ/b 56-Gb/s PAM-4 Receiver Using Enhanced Transition Utilization CDR and Genetic Adaptation Algorithms in 7-nm CMOS," *IEEE Solid-State Circuits Lett.*, vol. 2, no. 11, pp. 248–251, 2019.

[105] R. Schneider, "An Improved Pulse-Slimming Method for Magnetic Recording," *IEEE Trans. Magn.*, vol. 11, no. 5, pp. 1240–1241, September 1975.

[106] H. Zhang, E. Monaco, M. Bassi, and A. Mazzanti, "Flexible Transversal Continuous-Time Linear Equalizer Operating up to 25Gb/s in 28nm CMOS," in *2018 IEEE Int. Symp. on Circuits and Syst. (ISCAS)*. IEEE, 2018, pp. 1–4.

[107] A. Cortiula, M. Dazzi, M. Marcon, D. Menin, A. Bandiziol, A. Cristofoli, W. Grollitsch, R. Nonis, and P. Palestri, "A Simple and Fast Tool for the Modelling of Inter-Symbol Interference and Equalization in High-Speed Chip-to-Chip Interfaces," in *The 42nd Int. Conv. on Information and Communication Technol., Electronics and Microelectronics (MIPRO)*, Opatija, Croatia, May 2019, pp. 116–120.

[108] G.-S. Jeong, S.-H. Chu, Y. Kim, S. Jang, S. Kim, W. Bae, S.-Y. Cho, H. Ju, and D.-K. Jeong, "A 20 Gb/s 0.4 pJ/b Energy-Efficient Transmitter Driver Utilizing Constant-G_m Bias," *IEEE J. Solid-State Circuits*, vol. 51, no. 10, pp. 2312–2327, Oct. 2016.

[109] A. Bandiziol, W. Grollitsch, F. Brandonisio, M. Bassi, R. Nonis, and P. Palestri, "Design of a Half-Rate Receiver for a 10Gbps Automotive Serial Interface with 1-Tap-Unrolled 4-Taps DFE and Custom CDR Algorithm," in *2018 IEEE Int. Symp. on Circuits and Syst. (ISCAS)*, May 2018, pp. 1–5.

[110] "IEEE P802.3ch Multi-Gig Automotive Ethernet PHY Task Force," http://www.ieee802.org/3/ch/public/mar18/index.html, Mar. 2018.

[111] E. Di Biaso, B. Bergner, and C. Mandel, "High Speed Channel Modeling and Analysis," Apr. 2018. [Online]. Available: http://www.ieee802.org/3/ch/public/adhoc/Bergner_DiBiaso_Mandel_3ch_01_0418.pdf

[112] S. Ransom, "Jitter Analysis: The dual-Dirac Model, RJ/DJ, and Q-Scale," 2004. [Online]. Available: http://www.ece.tamu.edu/~spalermo/ecen689/jitter_dual_dirac_agilent.pdf

[113] A. Bandiziol, W. Grollitsch, F. Brandonisio, R. Nonis, and P. Palestri, "Design of a 8-Taps, 10Gbps Transmitter for Automotive Micro-Controllers," in *2016 IEEE Asia Pacific Conf. on Circuits and Syst. (APCCAS)*, Oct 2016, pp. 321–324.

[114] D. D'Ampolo, A. Bandiziol, D. Menin, W. Grollitsch, R. Nonis, and P. Palestri, "Automotive-Range Characterization of a 11 Gb/s Transceiver for Automotive Microcontroller Applications with 8-Tap FFE, 1-Tap Unrolled/3-Tap DFE and Offset-Compensated Samplers," in *2019 IEEE Asia Pacific Conf. on Circuits and Syst. (APCCAS)*. Bangkok, Thailand: IEEE, Nov. 2019, pp. 225–228.

[115] J. Lee, K. Park, K. Lee, and D.-K. Jeong, "A 2.44-pJ/b 1.62–10-Gb/s Receiver for Next Generation Video Interface Equalizing 23-dB Loss With Adaptive 2-Tap Data DFE and 1-Tap Edge DFE," *IEEE Trans. Circuits Syst. II*, vol. 65, no. 10, pp. 1295–1299, Oct. 2018.

[116] O. Elhadidy and S. Palermo, "A 10 Gb/s 2-IIR-Tap DFE Receiver with 35 dB Loss Compensation in 65-nm CMOS," in *2013 Symp. on VLSI Circuits*. IEEE, Jun. 2013, pp. C272–C273.

[117] A. Agrawal, J. F. Bulzacchelli, T. O. Dickson, Y. Liu, J. A. Tierno, and D. J. Friedman, "A 19-Gb/s Serial Link Receiver With Both 4-Tap FFE and 5-Tap DFE Functions in 45-nm SOI CMOS," *IEEE J. Solid-State Circuits*, vol. 47, no. 12, pp. 3220–3231, Dec. 2012.

[118] D. D'Ampolo, "Design and Characterization of Sampling Topologies with Adjustable Threshold for Fully-Adaptive Equalization in High Speed Serial Interfaces," Master's thesis, Università degli Studi di Udine, Udine, 2020.

[119] T. Shibasaki, W. Chaivipas, Y. Chen, Y. Doi, T. Hamada, H. Takauchi, T. Mori, Y. Koyanagi, and H. Tamura, "A 56-gb/s receiver front-end with a ctle and 1-tap dfe in 20-nm cmos," in *2014 Symp. on VLSI Circuits Dig. of Tech. Papers*, 2014, pp. 1–2.

[120] S. Krishnan and S. Pavan, "A 10 Gbps Eye Opening Monitor in 65nm CMOS," in *2015 IEEE Int. Symp. on Circuits and Syst. (ISCAS)*. IEEE, May 2015, pp. 3028–3031.

[121] M.-J. Park, H. Kim, M. Lee, and J. Kim, "Fast and accurate event-driven simulation of mixed-signal systems with data supplementation," in *2011 IEEE Custom Integrated Circuits Conf. (CICC)*, 2011, pp. 1–4.

[122] PCI-SIG, *PCI Express® Base Specification Revision 4.0, Version 1.0*, PCI-SIG Std., Oct. 2017. [Online]. Available: http://www.pcisig.com/specifications

[123] W. Lee, M. Shim, Y. Lee, H. Yang, H.-G. Ko, W.-S. Choi, and D.-K. Jeong, "0.37-pj/b/db pam-4 transmitter and adaptive receiver with fixed data and threshold levels for 12-m automotive camera link," in *ESSCIRC 2021 - IEEE 47th European Solid State Circuits Conf. (ESSCIRC)*, 2021, pp. 475–478.

[124] R. Sredojević and V. Stojanović, "Fully Digital Transmit Equalizer With Dynamic Impedance Modulation," *IEEE J. Solid-State Circuits*, vol. 46, no. 8, pp. 1857–1869, Aug. 2011.

[125] E. Groen *et al.*, "10-to-112-Gb/s DSP-DAC-Based Transmitter in 7-nm FinFET With Flex Clocking Architecture," *IEEE J. Solid-State Circuits*, vol. 56, no. 1, pp. 30–42, Jan. 2021.

[126] H. Ghafarian, S. Shivapakash, S. Mortazavi, P. Scholz, N. Lotfi, and F. Gerfers, "A 9-bit, 45 mW, 0.05 mm ^{2} Source-Series-Terminated DAC Driver With Echo Canceller in 22-nm CMOS for In-Vehicle Communication," *IEEE Solid-State Circuits Lett.*, vol. 4, pp. 10–13, 2021.

[127] Y. Lu, K. Jung, Y. Hidaka, and E. Alon, "Design and Analysis of Energy-Efficient Reconfigurable Pre-Emphasis Voltage-Mode Transmitters," *IEEE J. Solid-State Circuits*, vol. 48, no. 8, pp. 1898–1909, Aug. 2013.

[128] J. Kim, A. Balankutty, R. K. Dokania, A. Elshazly, H. S. Kim, S. Kundu, D. Shi, S. Weaver, K. Yu, and F. O'Mahony, "A 112 Gb/s PAM-4 56 Gb/s NRZ Reconfigurable Transmitter With Three-Tap FFE in 10-nm FinFET," *IEEE J. Solid-State Circuits*, vol. 54, no. 1, pp. 29–42, 2019.

[129] J. H. Jiang, S. Parikh, M. Lionbarger, N. Nedović, and T. Yamamoto, "A DC-46Gb/s 2:1 Multiplexer and Source-Series Terminated Driver in 20nm CMOS Technology," in *2014 IEEE Asian Solid-State Circuits Conf. (A-SSCC)*, 2014, pp. 377–380.

[130] W. Jung, J. Lee, K. Lee, H. Kim, and D.-K. Jeong, "A 8.4Gb/s Low Power Transmitter with 1.66 pJ/b using 40:1 Serializer for DisplayPort Interface," in *2020 Int. SoC Design Conf. (ISOCC)*, 2020, pp. 41–42.

[131] B. Razavi, "The StrongARM Latch [A Circuit for All Seasons]," *IEEE Solid-State Circuits Mag.*, vol. 7, no. 2, pp. 12–17, Jun. 2015.

[132] A. Bandiziol, W. Grollitsch, F. Brandonisio, R. Nonis, and P. Palestri, "Design of a Transmitter for High-Speed Serial Interfaces in Automotive Micro-Controller," in *2016 39th Int. Conv. on Information and Communication Technol., Electronics and Microelectronics (MIPRO)*, May 2016, pp. 84–88.

[133] F. Zhong *et al.*, "A 1.0625 $\sim$ 14.025 Gb/s Multi-Media Transceiver With Full-Rate Source-Series-Terminated Transmit Driver and Floating-Tap Decision-Feedback Equalizer in 40 nm CMOS," *IEEE J. Solid-State Circuits*, vol. 46, no. 12, pp. 3126–3139, Dec. 2011.

[134] C. Ahn, J. Hong, J. Shin, B. Kim, H.-J. Park, and J.-Y. Sim, "An 18-Gb/s NRZ Transceiver With a Channel-Included 2-UI Impulse-Response Filtering FFE and 1-Tap DFE Compensating up to 32-dB Loss," *IEEE Trans. Circuits Syst. II*, vol. 67, no. 12, pp. 2863–2867, Dec. 2020.

[135] I. Galton and C. Weltin-Wu, "Understanding Phase Error and Jitter: Definitions, Implications, Simulations, and Measurement," *IEEE Trans. Circuits Syst. I*, vol. 66, no. 1, pp. 1–19, Jan. 2019.

[136] B. Analui, J. Buckwalter, and A. Hajimiri, "Data-Dependent Jitter in Serial Communications," *IEEE Trans. Microw. Theory Techn.*, vol. 53, no. 11, pp. 3388–3397, Nov. 2005.

[137] W. Beyene, "Modeling and Analysis Techniques of Jitter Enhancement Across High-Speed Interconnect Systems," in *2007 IEEE Electrical Performance of Electr. Packaging*, 2007, pp. 29–32.

[138] F. Rao and S. Hindi, "Frequency Domain Analysis of Jitter Amplification in Clock Channels," in *2012 IEEE 21st Conf. on Electrical Performance of Electr. Packaging and Syst.*, 2012, pp. 51–54.

[139] J. Ardila and E. Roa, "On the Impact of Channel Loss on CDR Locking," in *2016 IEEE 59th Int. Midwest Symp. on Circuits and Syst. (MWSCAS)*, 2016, pp. 1–4.

[140] K. Kundert, "Predicting the Phase Noise and Jitter of PLL-Based Frequency Synthesizers," in *Phase-Locking in High-Performance Systems*. Wiley-IEEE Press, 2009, p. 736. [Online]. Available: https://designers-guide.org/analysis/PLLnoise+jitter.pdf

[141] M. van Ierssel, H. Yamaguchi, A. Sheikholeslami, H. Tamura, and W. W. Walker, "Event-Driven Modeling of CDR Jitter Induced by Power-Supply Noise, Finite Decision-Circuit Bandwidth, and Channel ISI," *IEEE Trans. Circuits Syst. I*, vol. 55, no. 5, pp. 1306–1315, Jun. 2008.

[142] C. Azeredo-Leme, "Clock Jitter Effects on Sampling: A Tutorial," *IEEE Circuits Syst. Mag.*, vol. 11, no. 3, pp. 26–37, 2011.

[143] Y. Choi, D.-K. Jeong, and W. Kim, "Jitter Transfer Analysis of Tracked Oversampling Techniques for Multigigabit Clock and Data Recovery," *IEEE Trans. Circuits Syst. II*, vol. 50, no. 11, pp. 775–783, Nov. 2003.

[144] N. Da Dalt, "Markov Chains-Based Derivation of the Phase Detector Gain in Bang-Bang PLLs," *IEEE Trans. Circuits Syst. II*, vol. 53, no. 11, pp. 1195 – 1199, 2006.

[145] N. Da Dalt, "Linearized Analysis of a Digital Bang-Bang PLL and Its Validity Limits Applied to Jitter Transfer and Jitter Generation," *IEEE Trans. Circuits Syst. I*, vol. 55, no. 11, pp. 3663–3675, Dec. 2008.

[146] M.-J. Park and J. Kim, "Pseudo-Linear Analysis of Bang-Bang Controlled Timing Circuits," *IEEE Trans. Circuits Syst. I*, vol. 60, no. 6, pp. 1381–1394, Jun. 2013.

[147] H. Xu and A. A. Abidi, "Design Methodology for Phase-Locked Loops Using Binary (Bang-Bang) Phase Detectors," *IEEE Trans. Circuits Syst. I*, vol. 64, no. 7, pp. 1637–1650, Jul. 2017.

[148] J. Liang, A. Sheikholeslami, H. Tamura, Y. Ogata, and H. Yamaguchi, "Loop Gain Adaptation for Optimum Jitter Tolerance in Digital CDRs," *IEEE J. Solid-State Circuits*, vol. 53, no. 9, pp. 2696–2708, Sep. 2018.

[149] G. Marucci, S. Levantino, P. Maffezzoni, and C. Samori, "Analysis and Design of Low-Jitter Digital Bang-Bang Phase-Locked Loops," *IEEE Trans. Circuits Syst. I*, vol. 61, no. 1, pp. 26–36, Jan. 2014.

[150] J. Sonntag and J. Stonick, "A Digital Clock and Data Recovery Architecture for Multi-Gigabit/s Binary Links," *IEEE J. Solid-State Circuits*, vol. 41, no. 8, pp. 1867–1875, Aug. 2006.

[151] M. Talegaonkar, R. Inti, and P. K. Hanumolu, "Digital Clock and Data Recovery Circuit Design: Challenges and Tradeoffs," in *2011 IEEE Custom Integrated Circuits Conf. (CICC)*, 2011, pp. 1–8.

[152] N. Da Dalt, "A Design-Oriented Study of the Nonlinear Dynamics of Digital Bang-Bang PLLs," *IEEE Trans. Circuits Syst. I*, vol. 52, no. 1, pp. 21–31, Jan. 2005.

[153] P. K. Hanumolu, G.-Y. Wei, and U.-K. Moon, "A Wide-Tracking Range Clock and Data Recovery Circuit," *IEEE J. Solid-State Circuits*, vol. 43, no. 2, pp. 425–439, 2008.

[154] W. T. Beyene, H.-S. Kang, A. Hashemi, G. Chen, and X. Liu, "Noise and Jitter Characterization of High-Speed Interfaces in Heterogeneous Integrated Systems," *IEEE Trans. Compon. Packag. Manuf. Technol.*, vol. 11, no. 1, pp. 109–117, Jan. 2021.

Short Biography

Davide Menin received the B.Sc. and M.Sc. degrees in electronics engineering from the University of Udine (Italy) in 2016 and 2018, respectively.

Since 2018 he has been a doctoral student on high-speed wireline transceivers for automotive applications at the University of Udine (Italy); his research interests related to SerDes include modelling and design of fully-adaptive equalization systems, modelling of CDR units and of system-level features of HSSIs.

List of Publications

Articles published in international journals:

- **D. Menin**, A. De Prà, A. Bandiziol, W. Grollitsch, R. Nonis, and P. Palestri, "A Simple Simulation Approach for the Estimation of Convergence and Performance of Fully-Adaptive Equalization in High-Speed Serial Interfaces," *IEEE Trans. Compon. Packag. Manuf. Technol.*, vol. 9, no. 10, pp. 2079–2086, Oct. 2019;

- **D. Menin**, T. Bernardi, A. Cortiula, M. Dazzi, A. De Prà, M. Marcon, M. Scapol, A. Bandiziol, F. Brandonisio, A. Cristofoli, W. Grollitsch, R. Nonis, and P. Palestri, "A Simple Modelling Tool for Fast Combined Simulation of Interconnections, Inter-Symbol Interference and Equalization in High-Speed Serial Interfaces for Chip-to-Chip Communications," *Advances in Science, Tech. and Eng. Systems J.*, vol. 5, no. 2, pp. 527–536, 2020;

- **D. Menin**, A. Bandiziol, W. Grollitsch, R. Nonis, and P. Palestri, "Design and Simulation of a 12 Gb/s Transceiver with 8-Tap FFE, Offset-Compensated Samplers and Fully-Adaptive 1-Tap Speculative/3-Tap DFE and Sampling Phase for MIPI A-PHY Applications," *IEEE Trans. Circuits Syst. II*, vol. 67, no. 8, pp. 1369–1373, 2020;

- P. Palestri, A. Elnaqib, **D. Menin**, K. Shyti, F. Brandonisio, A. Bandiziol, D. Rossi, and R. Nonis, "Analytical Modeling of Jitter in Bang-Bang CDR Circuits Featuring Phase Interpolation," *IEEE Trans. VLSI Syst.*, vol. 29, no. 7, pp. 1392–1401, 2021;

Papers presented at international conferences:

- A. Cortiula, M. Dazzi, M. Marcon, **D. Menin**, A. Bandiziol, A. Cristofoli, W. Grollitsch, R. Nonis, and P. Palestri, "A simple and fast tool for the modelling of inter-symbol interference and equalization in high-speed chip-to-chip interfaces," in *The 42nd Int. Conv. on Information and Communication Technol., Electronics and Microelectronics (MIPRO)*, Opatija, Croatia, May 2019, pp. 116–120;

- D. D'Ampolo, A. Bandiziol, **D. Menin**, W. Grollitsch, R. Nonis, and P. Palestri, "Automotive-Range Characterization of a 11 Gb/s Transceiver for Automotive Microcontroller Applications with 8-Tap FFE, 1-Tap Unrolled/3-Tap DFE and Offset-Compensated Samplers," in *2019 IEEE Asia Pacific Conf. on Circuits and Syst. (APCCAS)*. Bangkok, Thailand: IEEE, Nov. 2019, pp. 225–228;