DESIGNCON® 2022
WHERE THE CHIP MEETS THE BOARD

Validation Shift-Left: Enabling Early SerDes Mixed-Signal Validation

David Halupka, SeriaLink Systems
david@serialinksystems.com

Aleksey Tsychenko, SeriaLink Systems

Richard Allred, The MathWorks
Marc Erickson, The MathWorks
Tripp Worrell, The MathWorks
Barry Katz, The MathWorks

Jesson John, The MathWorks
Pragati Tiwary, The MathWorks

Venu Balasubramonian, Marvell Semiconductor
Lenin Patra, Marvell Semiconductor

Abstract

Modern mixed-signal ASIC designs, such as SerDes, require co-development of analog and digital subsystems that interact in increasingly complex ways. The challenges of today's silicon processes, as well as advancing performance targets, escalate analog design complexity. This, in turn, increases analog design time and introduces design cycle uncertainty. Moreover, analog design tends to lag digital design, which impedes digital validation and delays top-level mixed-signal validation.

For most mixed-signal designs, system models are used for architecture definition and design space exploration. In fact, analog specifications are derived from some form of system-level design exploration. Hence, the system models already embody the required analog functionality, configurability, and performance. We show that, with proper structuring, the system models can be used to automatically generate SystemVerilog models for analog components. Paired with netlistable pin definitions and synthesizable digital functionality, the core analog functionality from the system model can be used to construct run-time-configurable SystemVerilog analog models.

The availability of these automatically generated analog system models enables a shift-left of the validation effort to earlier in the workflow, such as during design exploration. This shift is accomplished by automating SystemVerilog model generation, while also allowing initial SystemVerilog models to be based on the architectural or design exploration models from which design specifications are derived. Furthermore, as the analog design evolves, and circuit simulation data becomes available, the system models can be refined and updated to match simulated analog behavior. This automation opens a path for re-generating updated SystemVerilog models throughout the project lifecycle: as the analog and digital designs evolve, implementation issues are addressed, and system-level design trade-offs are made and convergence is achieved.

The paper uses a common SerDes analog block, a continuous-time linear equalizer (CTLE), to demonstrate the proposed workflow. We begin with a white-board CTLE design, which is initially based on the 802.3ck reference COM receiver [1,2] CTLE specification. The COM-based CTLE model is augmented with a CTLE start-up digital calibration engine and the required digital control logic. The system model-to-SystemVerilog model export flow is then described, resulting in a design-specification-accurate CTLE model that can be used to shift-left design validation.

Next, a circuit model for the CTLE is presented and its circuit behavior is quantified using a targeted set of common analog circuit simulations. Finally, the simulation results are used to augment and refine the CTLE system model to match the circuit behavior. An updated, circuit-accurate, SystemVerilog model is automatically re-generated, reusing the presented model generation flow.

Author(s) Biography

David Halupka, Ph.D., is a co-founder of SeriaLink Systems. David has over 20 years of experience in mixed-signal and embedded system design. He was with Kapik for 11 years, where he served as Senior System-Architect and Principal Engineer. David also led the digital design team at Kapik. In 2018, he joined Intel's Mixed Signal-IP Group as Senior Systems Engineer, where he was responsible for adaptation algorithm development for the multi-standard SerDes. Ph.D, M.A.Sc., and B.A.Sc. from the University of Toronto.

Aleksey Tyshchenko, Ph.D., is a co-founder of SeriaLink Systems – a consulting team focusing on system modeling of high-speed serial links, IBIS-AMI modeling, model correlation, and system validation. SeriaLink Systems is working on building a configurable modeling flow to support SerDes projects through their entire life cycle: from architecture definition, through analog and digital design, to design validation. He has been working on behavioral modeling of high-speed SerDes systems, architecture analysis, adaptation, and signal integrity with multi-standard SerDes IP teams at V Semi and Intel. His Ph.D. research at the University of Toronto, Canada, focused on CDR systems for high-speed ADC-based receivers.

Richard Allred, Ph.D., is a principal software engineer at MathWorks, Inc., and has over a decade of signal integrity design experience at Intel, Inphi, and SiSoft. He has authored dozens of IBIS-AMI models and is currently developing signal integrity tools at MathWorks. M.S./B.S. 2006 and Ph.D. 2021 from the University of Utah.

Marc Erickson has spent his career in the design, verification, and tool development for large ASICs and FPGAs at Intel, Teradyne, and as a consultant. For the last 15 years he has been a technical lead at MathWorks making contributions for the HDL Verifier, SoC Blockset, SDR-Zynq, and Vision-Zynq products. BSEE from Princeton University.

Tripp Worrell joined MathWorks in 2017 after spending 3 years at SiSoft managing the development of their award-winning EDA simulation software. Prior to this he worked 7 years at Cisco Systems as a Signal Integrity and High Speed Design Engineer where he was responsible for design and verification of large enterprise networking linecards and backplanes. His current role as Development Manager overseeing SerDes Toolbox, Mixed-Signal Blockset and Signal Integrity Toolbox leverages both his hardware and software experience to best support the needs of MathWorks leading edge clients. Tripp received his BS in both Computer and Electrical Engineering (2005) and his MS in Computer Engineering (2007) from North Carolina State University.

Barry Katz, Director of Engineering, RF & AMS Products, leads the development teams responsible for RF, EM, Signal Integrity, SerDes, and Mixed-Signal modeling and simulation products at MathWorks. Prior to joining MathWorks, Barry served as President and CTO of SiSoft which he founded in 1995. At SiSoft, Barry was responsible for leading the definition and

development of SiSoft's products. He has devoted much of his career to delivering a comprehensive design methodology, software tools, and expert consulting to solve the problems faced by designers of leading-edge high-speed systems. Barry was the founding chairman of the IBIS Quality Committee. He received an MSECE degree from Carnegie Mellon and a BSEE degree from the University of Florida.

Jesson John is the analog mixed-signal segment manager at MathWorks. He works with semiconductor, electronics, and communications companies worldwide on improving design and verification workflows for AMS, SerDes, and signal integrity systems. Previously, he was a senior engineer and product development manager at Maxlinear, Inc., and Resonant, Inc respectively. Jesson has a master's degree in electrical and computer engineering from the University of Florida.

Pragati Tiwary, is a principal software engineer at MathWorks, Inc., and also team lead for Mixed-Signal Blockset product. Prior to this he worked 8 years at Cavium Inc. as a senior analog and mixed-signal design engineer. Pragati received his Bachelor of Engineering degree from Birla Institute of Technology (India) in Electronics and Communications Engineering (2005) and his MS in Electrical and Computer Engineering (2008) from Carnegie Mellon University.

Venu Balasubramonian is responsible for Marvell's high speed PHY product line that encompasses Ethernet transceivers from 10G to 800G. Venu has 20+ years of experience in the design, development and productization of high-speed interconnect products and various other types of wireline and wireless modems. He has 20+ issued patents in the area of signal progressing architectures and adaptive signal processing algorithms. He is an active participant at IEEE 802.3 standards groups with multiple contributions to 10G, 25G and 100G Ethernet standards. Venu has an M.Tech in Electrical Engineering from Indian Institute of Technology, Kanpur.

Lenin Patra is a Marvell Fellow and CTO for the Networking PHY Business Unit – overseeing Base-T Copper PHY, High Speed SerDes PHY and Optical PHY product line. Lenin has more than 20 years of experience building the product architecture for Networking applications which encompasses High Speed SerDes products along with MACSec and PTP, Optical PHY products for intra datacenter applications, Base-T Copper PHY products for Enterprise applications. He has multiple patents in networking field.

1 Introduction

Today's high-speed serial transceivers are complex systems that bring to bear the best a technology process has to offer to achieve 112 Gb/s transmission rates, and beyond. A SerDes transceiver leverages analog- and digital-signal processing to compensate for channel loss, reflections, and cross talk. For 112 Gb/s operation, most mid- to long-reach SerDes rely on ADC-based receivers [7,8], such as the one shown in Fig. 1. In an ADC-based receiver the continuous-time linear equalizer (CTLE) provides analog-signal processing, while the bulk of the signal processing is performed in the digital domain via the feed-forward equalizer (FFE) and the decision-feedback equalizer (DFE). The analog- and digital-signal processing paths are co-optimized to achieve the required performance targets, such as power, area, and link reach.

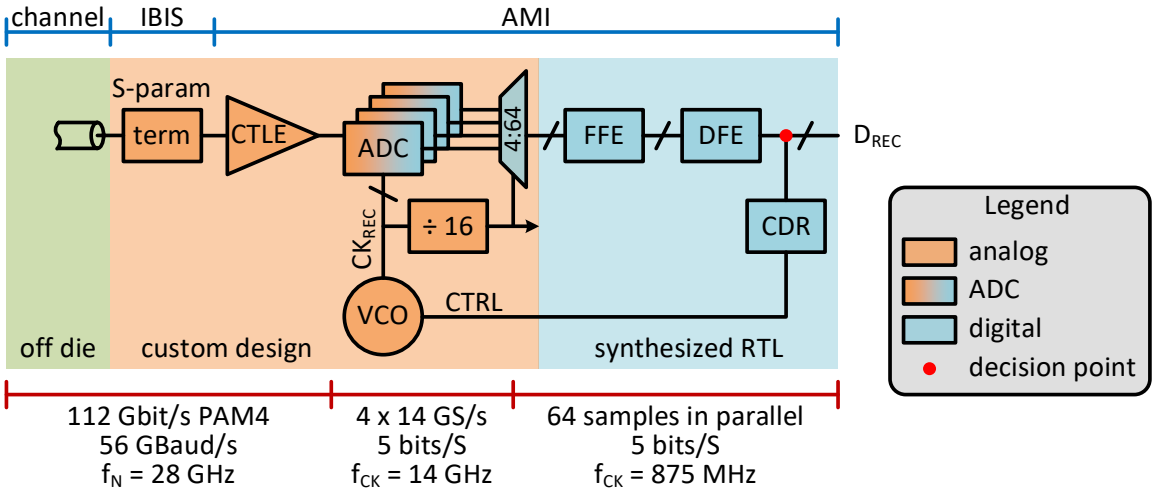


Fig. 1 – An ADC-based SerDes receiver consists of custom analog blocks (CTLE & VCO), mixed-signal blocks (ADC), custom digital blocks (4:64 demultiplexer), and synthesized digital (FFE, DFE, and CDR) working together to equalize channel losses and recover transmitted data.

The design of a high-speed serial transceiver is a monumental undertaking, requiring collaboration across a multi-disciplinary team, a multi-year development cycle, and a significant monetary investment. The multi-disciplinary nature of SerDes design requires involvement from system architects, analog designers, digital designers, layout designers, firmware engineers, signal integrity engineers, and other teams. Achieving a return on investment requires on-time delivery of a fully working and inter-operable transceiver in as few design iterations as possible – ideally one.

Designing and validating such a complex system necessitates the judicious use of models [10,12]. This paper specifically focuses on the use of models for overall system validation, including both system models that evaluate the end-to-end link performance, and models used for individual block functionality to validate a mixed-signal design. We will show that these behavioral models can be automatically generated earlier in the design cycle than with conventional modeling approaches. This enables a shift-left in the design validation effort. Section 2 introduces the different models that are used during a SerDes development cycle,

which represent the different levels of representation of the SerDes system. SerDes systems are complex mixed-signal systems, thus section 3 discusses the various mixed-signal validation challenges. We then discuss the typical behavioral model generation flows used, and then we propose an alternative methodology that allows for a left-shit for mixed-signal validation. Section 4 discusses modeling requirements to enable automatic behavioral model generation. In section 5 an architecturally accurate behavioral CTLE model is generated, which has the required input/output correspondence and is correlated to the underlying architectural model. The corresponding simplified circuit design for the CTLE is developed in Section 6, the architectural model is refined based on available circuit simulations, and an updated and circuit accurate behavioral model is automatically re-generated. Section 7 concludes the paper.

2 ABCs of SerDes Modeling: Model Use Cases

As with most complex mixed-signal systems, SerDes systems use various models throughout the development lifecycle to drive development and aid design decisions, as shown in Fig. 2. A SerDes development lifecycle consists of 3 major phases: architectural definition, circuit design and trade-off analysis, and finally validation. The models used for each activity can also be grouped into 3 corresponding categories: A-, B-, and C-models.

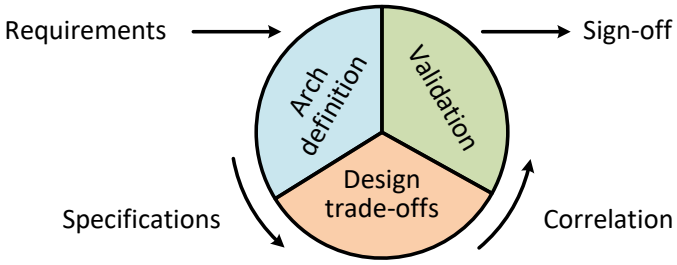


Fig. 2 – Typical SerDes design cycle can be represented on a circle. Requirements feed the architectural definition, which generates a set of specifications to be met during design. Design impediments may result in missed specifications, in which case design tradeoffs may need to be evaluated. Completed design are validated in isolation and in situ. In-situ validation requires the use of analog behavioral models for sign-off validation.

Architectural models (referred to in this paper as A-models) are used to evaluate potential architectures that can meet a given set of requirements. Architectural exploration concludes with a set of design specifications necessary to build the required architecture. Specifications are disseminated to the analog and digital teams for implementation. As the design evolves, the A-models should become more representative of the true circuit behavior, a necessary step towards performing trade-off analysis. An example of a trade-off analysis would be balancing out shortcomings in the analog-signal processing with improvements to the digital-signal processing.

Behavioral models (referred to in this paper as B-models) are used to substitute for incomplete or partially complete circuits to speed up simulations, or to enable simulations of the top-level design. B-models are critical for testing designs in situ. They are used by analog designers in the

form of behavioral circuit element models (substituting a bandgap reference generator with a fixed voltage source) or in the form of Verilog-A models to model analog and/or digital circuit behavior. Digital designers use B-models to emulate analog functionality, as digital logic either controls, interacts with, or processes information generated by analog blocks.

Circuit models (referred to in this paper as C-models[1]) are used by analog and digital design teams to enable circuit design exploration and block-level validation. These models help to determine if a set of design specifications are implementable or if design trade-offs are necessary. Design validation, as will be explained in Section 3, leverages circuit simulators and the underlying transistor device models, with a focus on validating the individual blocks in isolation.

SystemVerilog is a popular modeling language used for generating behavioral (numerical) models for analog functionality. While the syntaxes of SystemVerilog and Verilog-A are similar, these languages are intended to be used with different simulators. Verilog-A is targeted at continuous-time circuit simulators, where the simulator can change the time-step dynamically depending on specified relative or absolute simulation tolerances. Verilog-A is a useful analog-centric language that can be used for modelling continuous-time systems; it is usually used to model small circuits or elements. SystemVerilog is targeted at digital simulators, where the simulator advances time based on signal change events or fixed-time steps.[2]

B- and C-models need to correlate with each other because top-level design validation is carried out using B-models, for analog functionality, using a digital simulator. Design teams use some form of top-level testing as final sign-off criteria for chip fabrication. Therefore, B-models need to be shown to correlate to the C-model (circuit) behavior.

To achieve design sign-off, the completed design must ultimately operate as specified, achieve the desired performance targets, and be production-ready in as few design iterations as possible – ideally one. Achieving first-time correct functional silicon requires sufficient verification of the complete mixed-signal design: not just within each domain but across all design domains, and not just before manufacturing release (tape-out) but throughout the project lifecycle. Thus, mixed-signal validation is critical for SerDes mixed-signal designs. The next section reviews mixed-signal validation challenges.

3 Mixed-Signal Validation Challenges

A mixed-signal system consists of an intermix of analog and digital blocks that must work seamlessly together [11]. To do so, first the analog and digital blocks must be verified independently and in isolation; then they must be validated to work together, necessitating

[1] Not to be confused with the C programming language. In this paper, C-model refers to a circuit based model: analog circuit design or RTL digital logic design.
[2] Verilog-AMS is a language that supports Verilog-A and SystemVerilog syntaxes. Verilog-AMS is intended to be run in a mixed-signal simulation environment where the discrete-time parts of the Verilog-AMS code are simulated by a digital simulator and the analog parts by an analog simulator.

mixed-signal validation. A true mixed-signal simulation uses a digital simulator for the digital components and an analog simulator for the analog components. The two simulators communicate with each other, exchanging signal information across the analog/digital domain boundary. These mixed-signal simulations are notoriously slow, mainly because analog simulations are computationally expensive and because keeping the two simulators in lockstep while providing the implicit DAC and ADC functionality between the two simulators requires considerable overhead [10]. Moreover, a mixed-signal simulation requires a complete analog design, and completion times are increasingly difficult to predict.

Nanoscale processes are not analog design-friendly, and transistor layout-dependent effects can have a significant impact on analog circuit performance, requiring re-validation and potential re-design as layouts are completed. These design difficulties and increasing performance targets increase analog design completion time and uncertainty. The simulation time required for analog validation is yet another factor. Analog simulations are computationally expensive but are necessary to fine-tune and guarantee analog performance over all operating conditions and process corners. Analog simulators offer a multitude of analysis options for analog validation, and although computationally expensive time-domain simulations are used very sparingly, mixed-signal simulations necessitate time-domain simulations. This makes using actual analog circuits in mixed-signal simulations difficult because of the computational overhead and late delivery of analog designs.

On the other hand, digital simulations are fast, but the configuration space is large, and because of the presence of finite state machines, the operational space has non-linear discontinuities that must be explored and validated. To be functionally validated, digital circuits require an input stimulus; in SerDes systems that stimulus comes from analog circuits, such as from a clocked comparator or an ADC, and this input must be time-domain simulated.

Although the behavior, interfaces, and control ports of analog blocks can be captured and described by design specifications, it is possible for subtle details to be lost in communication: as an example, the polarity of differential signals is a common point of miscommunication. Correlated analog behavioral models are required both for digital design and validation. SystemVerilog analog mixed-signal models are used for analog block functionality to accelerate mixed-signal validation and to bridge the gap between digital- and analog-design completion.

3.1 Typical B-model creation flows

A B-model for an analog block can be written by a design engineer or created using an automated flow; the advantages and disadvantages for each approach are summarized in Table 1. A digital engineer can create B-models, but may not be capable of capturing subtle, yet critical, analog behavior. An analog engineer, while very aware of subtle analog behaviors, may not be fluent in Verilog and may be unable to accurately capture the required behavior; moreover, using an analog engineer to write B-models is counterproductive to finalizing and validating the analog design. In either case, human-created B-models are intrinsically prone to transcription and description errors, and thus should be cross-checked against the analog design

itself, which thereby introduces overhead and additional model validation. Commercial tools to automatically create B-models are available but require a complete analog design (C-model) to characterize.

Table 1 - Comparison of different B-model generation methods

Method	Advantage	Disadvantage
Digital Engineer	Proficient in Verilog	May not capture important yet subtle analog behavior
Analog Engineer	Able to generate detailed model	Not Verilog-proficient nor aware of simulator limitations; impedes analog design progress
B-from completed C-model	Automatic model generation	Analog design must be finalized; requires additional simulation overhead

3.2 B-models from A-models

SerDes designs leverage system modeling for architectural design exploration and evaluation of the expected link performance. Initially, these A-models encapsulate the required analog and digital functionality. As the design progresses, the A-models are updated to reflect the changes and tradeoffs needed during design implementation. The initial A-models used for design exploration are used to drive analog and digital specifications and requirements; as such, these initial A-models embody the analog and digital design requirements.

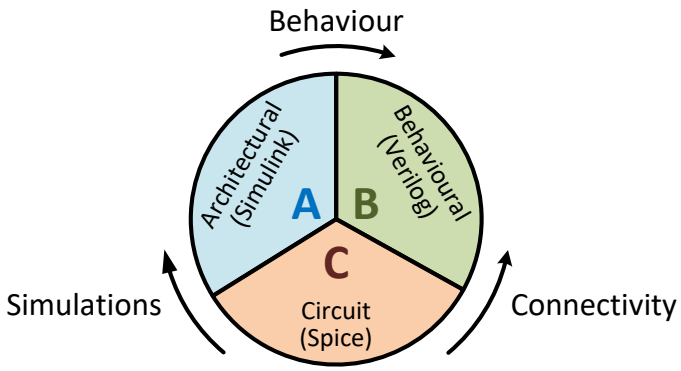


Fig. 3 – The design phases outlined in Fig. 2 use different models. A-models are used for design space exploration. C-models are used to verify that design specification targets have been met. Missing a design target requires a trade-off analysis to be made by providing implementation feedback into the A-models, which as a result become more representative of the true circuit functionality. B-models are used as stand ins.

Rather than manually creating B-models, or automatically creating them upon analog design completion, this paper proposes automatically creating B-models based on A-models as they are incrementally refined during the design process, as shown in Fig. 3. Up-to-date A-model functionality is integrated with connectivity information from the C-model hierarchy to create functional B-models. Initially, the B-models is based on the desired analog functionality encapsulated by the A-models, allowing one to shift-left the mixed-signal validation effort. As the analog design matures and the A-models are refreshed the updated circuit-accurate B-

models are re-exported based on these refined A-models. The A-models are refreshed based on simulation-based characterization data (simulations arrow in Fig. 3) as it becomes available.

4 Enabling Automatic B-Model Generation

SystemVerilog is limited in the complexity of analog functionality that can be represented using the language's constructs and syntax. However, Verilog simulators support a direct programming interface (DPI) that enables the use of custom languages (C, C++, etc.) to provide increased modeling flexibility. Nonetheless, C and C++ are low-level languages and require programing all required analog behavior as well as providing a programming interfacing to Verilog simulator API – a high-barrier to entry. On the other hand, A-models are described in high-level modeling languages such as MATLAB or Simulink, which offer high-level support packages in the form of Toolboxes and Blocksets.

MATLAB and Simulink have supported export flows to SystemVerilog DPI since R2014b. These capabilities enable design teams to leverage existing high-level system models (A-models) to create B-models, thereby allowing them to shift-left mixed-signal validation. However, to take advantage of this flow, a few A-model requirements must be met.

4.1 Requirement: Matched hierarchy

To use existing A-models, the hierarchy of the A-model should match the C-model design hierarchy. The A-model hierarchy does not need to match the design hierarchy exactly, but rather only to the level at which a B-model export is desired. For example, a CTLE may consist of multiple stages, but the B-model will only be exported for the whole CTLE; thus, modeling each stage independently may not be necessary. As another example, an ADC implementation may consist of multiple analog blocks, such as amplifiers, active filters, etc., yet this level of detail may not be required for the B-model, nor for the A-model.

4.2 Requirement: Stable interfaces

For the A- and B-models to co-evolve with the design, the input/output interfaces for the system models need to be fixed. It is likely that the A-model uses a subset of the circuit level interfaces – for example, test port interfaces may be omitted – and this is not an issue, as will be shown later. The important thing is that the interface should be relatively stable throughout the project lifecycle, as is often the case because interfaces are defined as part of the design specification: inputs, outputs, controls, and test ports as required.

4.3 Requirement: Fixed time-step implementation

Time, as dictated by the Verilog simulator, is strictly increasing; hence, unlike a continuous-time simulator, the simulator does not check signal node amplitude changes, decrease time steps, nor go back in time and re-evaluate signal node changes. As such, B-models used for Verilog simulations need to be fixed-time step models; they need to be discrete-time systems that model continuous-time behavior. The DPI-based B-models generated by MATLAB/Simulink are

dynamic linked library (DLL) modules created from fixed-time step MATLAB System objects. They are integrated into the Verilog simulation by triggering their execution based on periodic events, like a rising clock edge or a time-based trigger. The exception is that a stateless, purely combinatorial design is not sensitive to time and can be triggered to execute on any input value change instead.

4.4 Enabler: Object-oriented system models

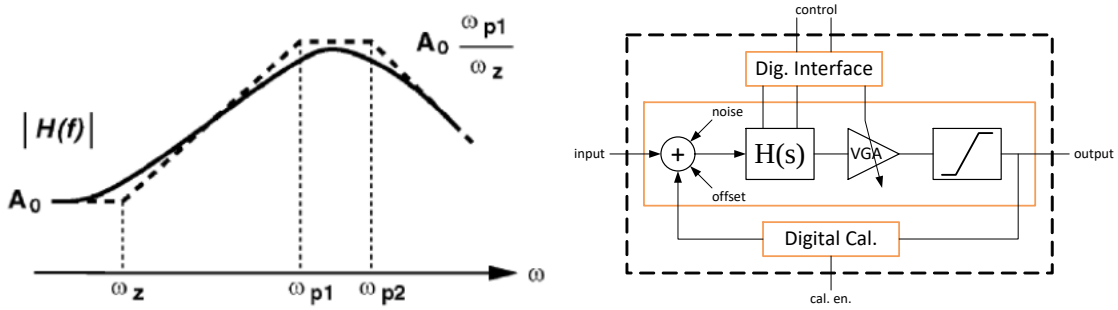
An object-oriented modeling approach simultaneously meets both requirements: the interface can be defined early in the design process, and the internals of the A-model can change as the C-model matures. An object-oriented approach also allows one to encapsulate the required fixed time-step solver into the model itself, which also enables selection of the accuracy of the model's solver depending on its usage: low-precision functionality for connectivity-only tests, and higher-precision functionality for link calibration validation, link bring-up, or system functionality tests.

The next section demonstrates the process of generating a B-model from an object-oriented CTLE A-model. This A-model will be subsequently refined by using C-model simulation results to refine the internals of the A-model to represent the simulated C-model behavior more faithfully.

5 B-Model Generation: Behavioral CTLE

The CTLE is a common analog block used in SerDes systems; it provides high-frequency amplification and low-frequency attenuation to counteract the channel's high-frequency attenuation. The CTLE may consist of multiple stages and may also provide broad-band amplification via a variable gain amplifier (VGA). For simplicity, this paper will be limited to constructing a B-model of a single-stage CTLE; however, the approach and steps can be easily extended to modeling a multi-stage CTLE.

During architectural exploration, the CTLE may be defined based on guidance from the channel operating margin (COM) reference model [1,2], which specifies only its frequency domain response, as shown in Fig. 4(a). The COM-specified CTLE supports a programmable (tunable) frequency response, allowing for control of the amount of high-frequency boost applied to the



(a) (b)

Fig. 4 - (a) CTLE frequency response is defined by the peak frequency (ω_{p1}) and the boost amplitude (A_0). (b) Simplified CTLE model with boost + VGA stage and digital controls.

incoming signal. However, the C-model may be further specified to also control the location of the mid-band zero, and potentially offer additional controls as well.

A conceptual B-model block diagram for the CTLE is shown in Fig. 4(b), wherein the applied input is first summed with the expected input-referred circuit noise, as well as the expected input-referred differential offset. The resulting signal is then passed through a controllable frequency-domain filter, which represents the small-signal (AC) response of the filter. The filter is then followed by a VGA, before being subjected to a limiting amplifier that models the expected large-signal compressive behavior of the CTLE. The control signals, which set and control the parameters of the CTLE, may need to be first conditioned by a digital interface (for example, to convert binary-weighted control signals into thermometer controls). Finally, a digital calibration block may be included within the CTLE or offer external observation ports and control ports to the CTLE, allowing for trimming of the additive intrinsic differential input offset.

The A-model that is exported into a B-model, is shown in Fig. 5. The A-model consists of a tunable one-stage CTLE, with an expected output-limiting saturating behavior. The tunable CTLE stage is modeled using the SerDes Toolbox™ CTLE System Object™, whereas the output swing compression is modeled using the saturating amplifier system object.

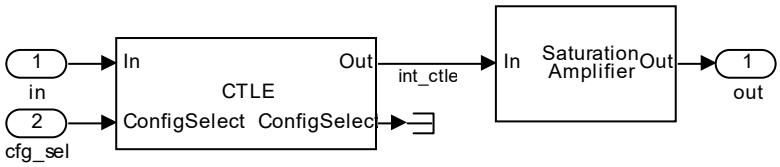


Fig. 5 - A Simulink® CTLE behavioral model consists of a SerDes Toolbox™ CTLE system object, as well as the VGA and saturating amplifier system objects.

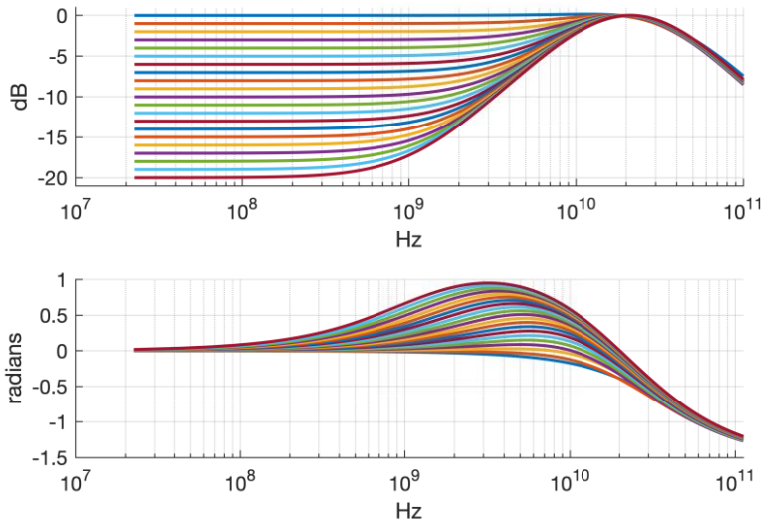


Fig. 6 - COM CTLE magnitude transfer function: peak boost occurs at 20 GHz and supports DC attenuation of 0 to -20 dB.

The CTLE is specified by providing the peaking gain frequency (20 GHz), and the ranges of the DC gain and boost: -20 to 0 dB and 0 to 20 dB, respectively. The resulting gain and phase responses are shown in Fig. 6.

The behavioral saturating amplifier, which models the expected finite output swing of the C-model due to voltage headroom limitations, is specified by its linear gain and one-sided peak swing, 1 V/V and 0.4 V, respectively. The resulting non-linear distortion applied to the CTLE

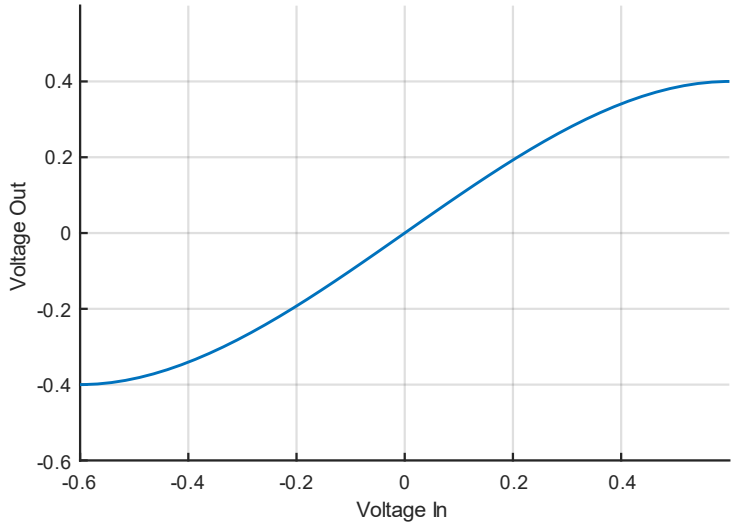


Fig. 7 – Saturating amplifier's DC input-to-output transfer function.

output is shown in Fig. 7.

5.1 Interface definition

As illustrated in Fig. 3, the interface definition comes directly from the C-model: interfaces are defined early in the project cycle, based on the design specifications and the resulting A-model-driven design hierarchy. However, the A-model's interface is unlikely to match the C-model's interface exactly. For example, the CTLE C-model may have differential inputs and outputs, whereas the A-model may use single-ended inputs and outputs that represent the C-models differential signal magnitudes, as in this paper.

Discrepancies between a C-model's interface and the simplified A-model interface can be reconciled by using an A-to-B-model interface gasket. This gasket, as illustrated in Fig. 8, can perform the requisite differential-to-single-ended signal conversion at the input and single-ended-to-differential conversion at the output. Random, or programmable, offset can also be added by the interface gasket into the signal path, as can the offset calibration compensation. The core functionality of the CTLE (ostensibly the difficult-to-model component) is generated directly from the A-model. The next section describes how the various pieces of the B-model are automatically connected during the A-to-B-model export flow.

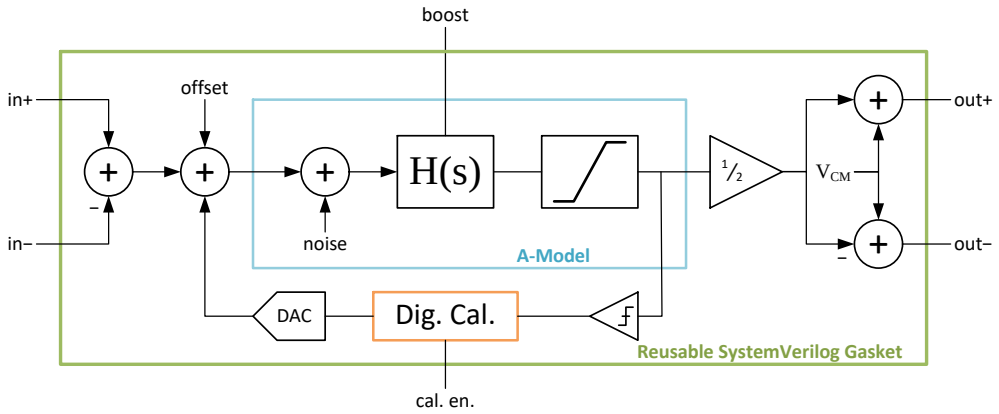


Fig. 8 - CTLE B-model block diagram. Notice that the core analog functionality is encapsulated by the A-model. A reusable SystemVerilog gasket provides B-to-A model interfacing while pulling in additional support modules that are not present in the A-model: such as the clocked comparator, digital calibration engine, and DAC.

5.2 Missing A-model functionality

The A-model may purposely omit low-level functionality, such as calibration circuits. These circuits exist in the C-model and provide sensing and actuating points that may be driven externally or by a self-contained calibration circuit. For example, in the CTLE B-model shown in Fig. 4(b), the digital calibration block is used to compensate for the input offset of the CTLE. This calibration circuit runs upon power-up and has no effect on the signal processing path after the input offset is compensated, making it immaterial and invisible to the A-model. However, the B-model may need to capture this behavior because start-up calibration is a potential top-level sign-off simulation.

Additional functionality necessary for the B-model can also be incorporated by the A-to-B-model gasket. For example, the digital calibration block and associated DAC and comparator will also be included in the overall CTLE B-model by the A-to-B-model gasket. The functionality of the DAC, calibration engine, and sampler can come from a separate A-to-B-model export flow by directly pulling in synthesizable RTL, or by resorting to low-level SystemVerilog modeling.

The digital calibration feedback loop consists of a clocked comparator running at a locally generated frequency, a digital calibration engine, and a DAC to apply the offset to the CTLE input. When the calibration feedback loop is enabled and the inputs to the CTLE are shorted, the calibration engine integrates the sampled CTLE output and drives the loop in feedback to cancel out any input offset.

For the example CTLE, the digital calibration engine is described in synthesizable RTL, and as such can be directly included into the B-model as a module. The digital calibration engine details are shown in Fig. 9. This engine is enabled by the cal_en signal. The engine consists of a 1st-order integrator that integrates the ±1 clocked comparator outputs into a multi-bit digital value. The upper MSBs of this digital value in turn drives the offset compensation DAC. When the offset is nullified, the counter value will dither around a steady-state value, at which point the offset should be within the single LSB resolution of the DAC.

For the CTLE example, the DAC is emulated in the B-model using a constant scaling factor, which converts the 2's compliment binary number conversion into an offset compensation voltage. However, the DAC could also have been modeled using an exported DAC MATLAB or Simulink model. With all of the B-model components accounted for, let us turn our attention towards the model export flow.

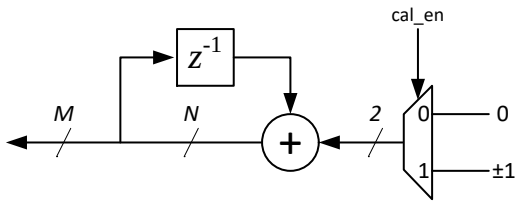


Fig. 9 – Digital offset calibration engine. When enabled, the sampled (high/low) CTLE differential output is digitally integrated using an N-bit up/down counter. The upper M bits of the counter are used to drive a DC offset DAC, which provides a constant DC offset into the CTLE that compensates for its intrinsic offset.

5.3 Model export

The model is exported using the Simulink export flow [6]: via HDL Verifier™ or Simulink® Coder™ and HDL Verifier™. This flow allows for the use of a Verilog export template, which can be customized to call the exported DLL as required. The template used for export is shown in Fig. 10(a): it specifies that the CTLE DLL is to be called periodically, as set by the time_step parameter. The CTLE is initialized upon simulation start. The CTLE model's update and output functions are called every sample interval, which is a fraction of the symbol interval. The update function takes the current input as its argument, and the output function provides the corresponding output for the sample interval step. The resulting exported CTLE DPI model wrapper is shown in Fig. 10(b).

```
// CTLE DPI model

`timescale 1s / 1ps

module %<FileName> (
   %<PortList>
);

localparam time_step = 1/28e9/32;

%<ObjHandle>
%<ImportInitFunction>
%<ImportOutputFunction>
%<ImportUpdateFunction>
%<ImportSetParamFunction>

initial begin
  %<CallInitFunction>
  forever begin
    #(time_step)
    %<CallUpdateFunction>
    %<CallOutputFunction>
  end
end

endmodule
```

(a)

```
// CTLE DPI model

`timescale 1s / 1ps

module CTLE_dpi (

/* Simulink signal name: 'ctle_in' */
input real ctle_in ,
/* Simulink signal name: 'cfg_sel' */
input byte unsigned cfg_sel ,
/* Simulink signal name: 'gain' */
input real gain ,

/* Simulink signal name: 'out' */
output real out
);

localparam time_step = 1/28e9/32;

chandle objhandle;
import "DPI-C" function chandle DPI_CTLE_initialize(chandle existhandle);
import "DPI-C" function void DPI_CTLE_output(input chandle objhandle,
/*Simulink signal name: 'ctle_in'*/
input real ctle_in,
/*Simulink signal name: 'cfg_sel'*/
input byte unsigned cfg_sel,
/*Simulink signal name: 'gain'*/
input real gain,
/*Simulink signal name: 'out'*/
inout real out);
import "DPI-C" function void DPI_CTLE_update(input chandle objhandle,
/*Simulink signal name: 'ctle_in'*/
input real ctle_in,
/*Simulink signal name: 'cfg_sel'*/
input byte unsigned cfg_sel,
/*Simulink signal name: 'gain'*/
input real gain);


initial begin
  objhandle = DPI_CTLE_initialize(objhandle);
  forever begin
    #(time_step)
    DPI_CTLE_update(objhandle,
ctle_in, cfg_sel, gain);
    DPI_CTLE_output(objhandle,
ctle_in, cfg_sel, gain, out);
  end
end

endmodule
```

(b)

Fig. 10 – B-model (a) export template and the resulting (b) SystemVerilog DPI model generated.

The CTLE model wrapper used is shown in Fig. 11. This A-to-B-model wrapper provides the require A-to-B-model signal conditioning and instantiates the C-model components that may not be present in the A-model. This wrapper provides all the functionality outside of the blue box in Fig. 8. Having generated the B-model from the A-model, we now show that the B-model behavior is correlated to the A-model and that the B-model can be used to verify the operation of the digital offset calibration engine.

```
`timescale 1s / 1fs

module ctle #(
   parameter offset = 0.0,
   parameter cm = 0.5
) (
   input arst_b, // Asynchronous reset
   input cm_clk, // common-mode compensation clock
   input cm_cal_en, // common-mode calibration enable
   input real inp, // CTLE differential input +
   input real inm, // CTLE differential input -
   input byte unsigned boost, // boost setting
   output real outp, // CTLE differential output +
   output real outm  // CTLE differential output -
);

   real in, offset_comp, out;
   reg out_q;
   wire signed [7:0] offset_comp_dig;

   always @(*) begin: CTLE_step
      offset_comp = offset_comp_dig * 0.125/128; // offset DAC model
      in = inp - inm + offset - offset_comp;

      outp = cm + out/2;
      outm = cm - out/2;
   end

   // Clocked comparator
   always @(posedge cm_clk)
         out_q <= cm_cal_en? outp > outm: 1'b0;

// CTLE digital offset calibration
ctle_cm_cal #(.out_width(8), .int_width(12)) ctle_cm_cal (.arst_b(arst_b), .clk(cm_clk),
   .en(cm_cal_en), .sense(out_q), .comp(offset_comp_dig));

// CTLE B-model core (based on exported A-model)
CTLE_dpi ctle_core(.ctle_in(in), .cfg_sel(boost), .out(out));

endmodule
```

Fig. 11 – CTLE B-to-A-model gasket, which provides all the A-model missing components and the input and output interface translation required for the B-model illustrated in Fig. 8.

5.4 A-to-B correlation results

The Verilog simulator supports only transient simulations; therefore, any comparison of A-to-B-model behavior must be based on transient simulations. As the CTLE is a linear filter, it can be fully characterized using an impulse response. A step response works equally well and has the added benefit of being easier to simulate in a fixed-step simulator.

A step response is applied to both the A- and B-models in their respective simulators. The resulting family of step responses is shown in Fig. 12(a) for the A-model and in Fig. 12(b) for the

B-model. Notice that the two sets of curves are well-matched. The position of the 0.2-V input step is denoted by the dashed red line in both plots.

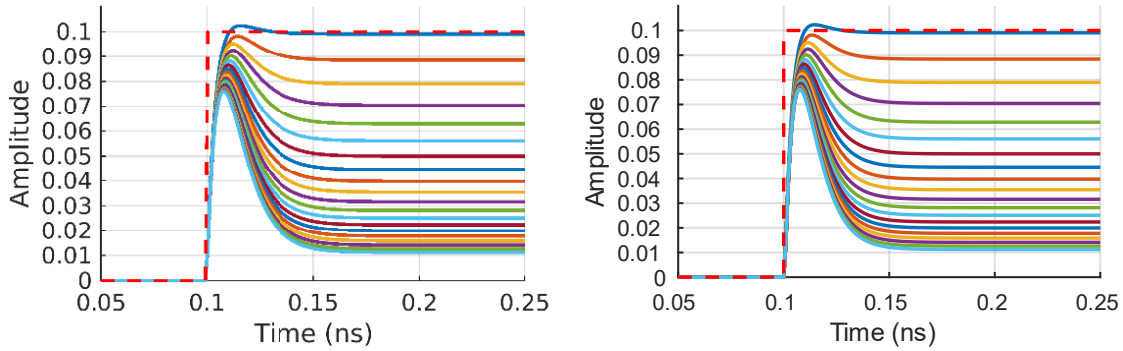


Fig. 12 - Step responses as measured in (a) Simulink for the A-model and (b) Verilog for the B-model.

The step responses shown in Fig. 12 exercise the filter part of the CTLE, and the output swing compression modelled by the saturating amplifier can be seen by looking at low- and high-amplitude PRBS patterns. A scaled PRBS-7 input pattern is applied to the A- and B-model CTLEs, and the peak amplitude of the PRBS pattern is either 100 or 400 mV. The output waveforms for the A- and B-model CTLEs are shown in Fig. 13 for (a) minimum and (b) maximum CTLE boost settings. The top plots show the A- and B-model transient responses for a 100-mV PRBS sequence, whereas the bottom plots show the response for a 400-mV PRBS sequence. In all four cases, the B-model output correlates exactly with the A-model output; thus, the B-model is well-correlated to the A-model.

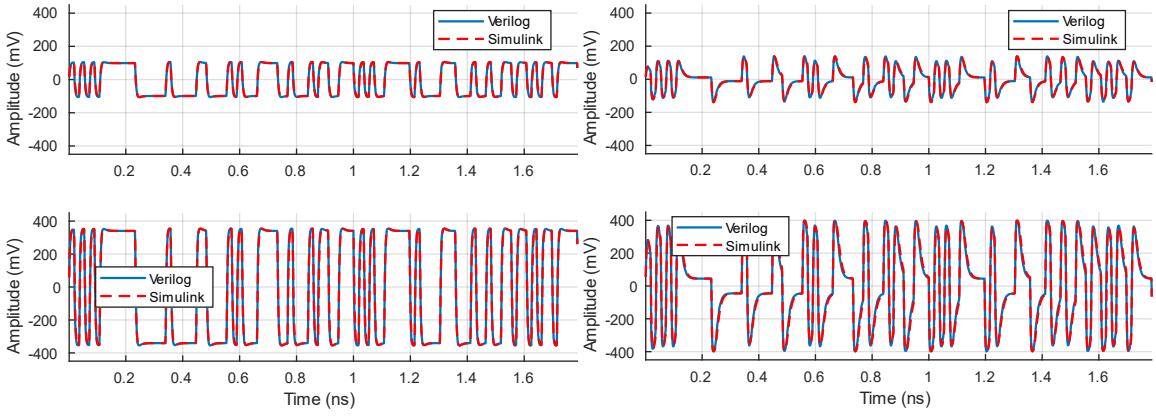


Fig. 13 – Verilog and Simulink output waveforms for behavioral CTLE driven by a PRBS waveform with a (top) 100-mV and (bottom) 400-mV amplitude for (a) minimum and (b) maximum boost settings.

5.5 Validating digital calibration engine using B-model

Early access to B-models provides another benefit to mixed-signal designs: the ability to left-shift the development and validation of closed-loop analog-digital control systems. The B-model CTLE example generated in the previous section, which is based on the COM specification of the required CTLE behavior, allows for the early testing of the digital offset compensation controller.

In fact, the initial offset compensation controller developed for the CTLE example in this paper initially had a design flaw – a signal polarity inversion – that was found and rectified using the B-model version of the CTLE. Hence, not only can development of digital assist be left-shifted, but so too can the development of testbenches and pass/fail criteria.

The closed-loop behavior of the digital offset compensation controller is shown in Fig. 14. The top part of the figure shows the differential outputs of the B-model CTLE converging towards the output common mode voltage of 0.5 V. The bottom portion of Fig. 14 shows the differential output voltage of the CTLE, and the compensation DAC output.

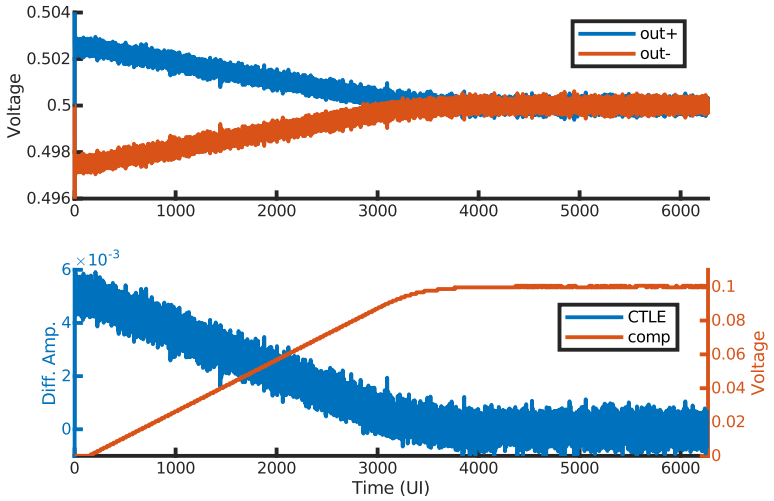


Fig. 14 – The digital offset calibration engine showing (top) the differential output of the CTLE, and (bottom) measured differential output voltage (blue) and DAC output voltage applied to compensate for the offset (red).

Although the CTLE and associated digital offset cancellation controller is a rather simple example, and the digital control loop is straightforward, the B-model generation flow can be applied to much more complicated applications. Moreover, as the C-model matures, the A-model can be refreshed with up-to-date simulation characterizations, allowing for the exact same export flow to be re-used to generate a more accurate B-model. In the next section, a C-model version of the CTLE is developed and simulated using a circuit simulator, and the simulation characterizations are used to refine the A-model, and in turn regenerate an updated B-model.

6 Circuit-Accurate A- and B-Models

The example CTLE C-model is intended to illustrate the process of using C-model simulations to refresh the A-model, and in turn automatically update the B-model as well. The CTLE design presented here is a simplified embodiment of a manufacturable CTLE. This simplified CTLE circuit is used to show (1) some of the required simulations analog designers need to run to characterize the CTLE; (2) how data from these simulations is processed and used to refresh the A-model; and (3) how the updated A-model can then be exported into a circuit-correlated B-model. We begin with an overview of the CTLE analog circuit.

6.1 CTLE analog circuit

The CTLE topology used here is the resistor-capacitor source-degenerated differential amplifier, shown in Fig. 15. The source degeneration network provides control of the boost frequency and magnitude: the capacitor sets the frequency of the zero, and the resistor sets the DC attenuation. This design is a simplified circuit implementation of a CTLE; the differential pair transistors serve as the main circuit impediments allowing us to showcase the effects of (1) the limited voltage headroom, which manifests itself as differential signal compression; (2) the high-frequency gain drop due to parasitic transistor capacitances; and (3) the measurable differential offset due to transistor mismatch. Next, the CTLE circuit is characterized within a circuit

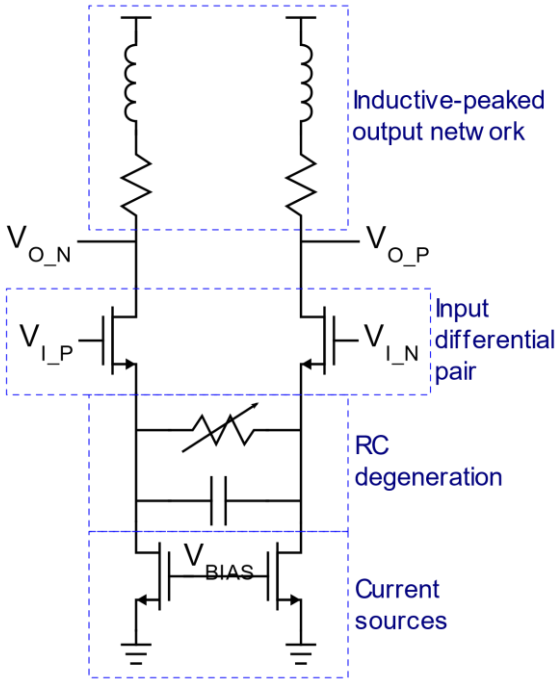


Fig. 15 - Simplified CTLE circuit.

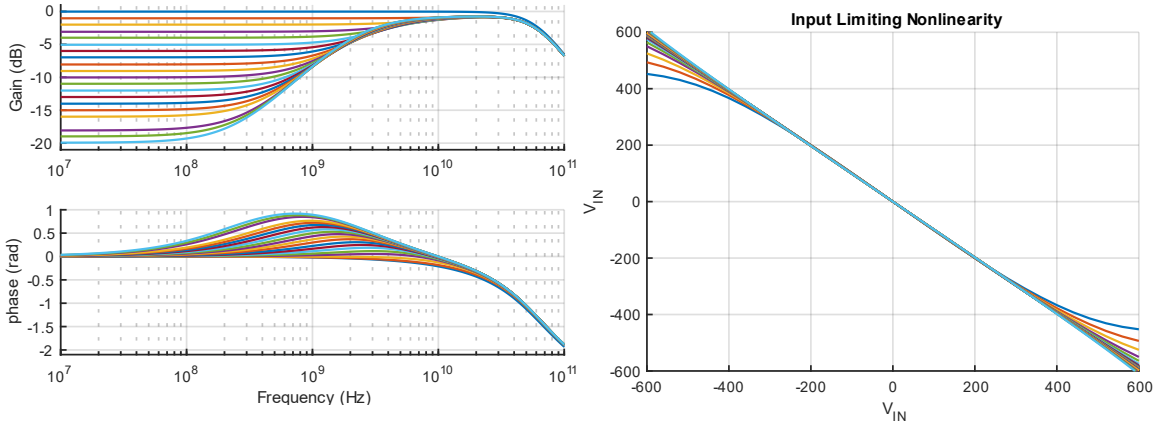
simulator.

6.2 Required analog simulation data

To convert the CTLE A-model from a specification-based model to a more circuit-representative model requires that the C-model be characterized using a circuit simulator. The set of measurements required to characterize the CTLE are very similar, if not exactly the same, to the set of simulations an analog designer would use to evaluate the circuit performance, including:

- an AC (small signal) frequency-domain analysis of the circuit's input-to-output transfer function
- a DC (large signal) voltage-based analysis of the circuit's input-to-output transfer function
- a Monte-Carlo analysis of the circuit's differential output, given a 0-V common-mode centered input.

The simulated AC response of the C-model is shown in Fig. 16(a) for all boost settings. The measured C-model response differs from the COM-like A-model response shown in Fig. 6: the desired peaking frequency is the same, but the boost bandwidth is wider. In this case, this discrepancy is intentional to highlight the fact that a C-model implementation may not be able to exactly match the expected (specified) behavior prescribed by the A-model.



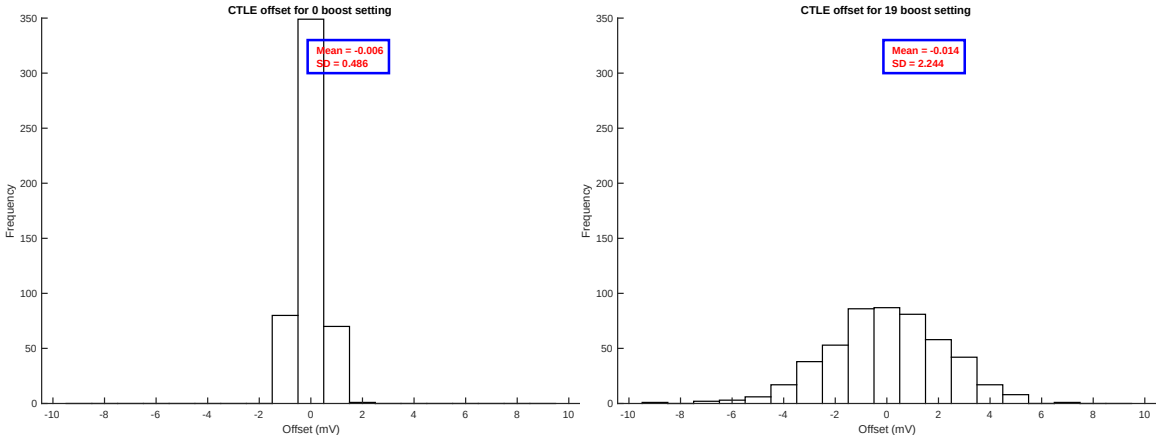
(a) **(b)**

Fig. 16 - Circuit measured CTLE (a) boost-dependent gain and phase response, and (b) input-limiting non-linearity.

The C-model's DC transfer characteristics illustrate the input- or output-limiting (compressive) behavior of the circuit; that is, the range of input or output voltage that causes the circuit to depart from its desired linear behavior. The simulated input-limiting behavior is shown in Fig. 16(b) for all boost settings.

Manufacturing variation, including random transistor threshold voltage variation, will cause the manufactured CTLE C-model to have a finite input-referred offset. Left uncompensated, this input-referred offset will be reflected in the output of the CTLE as a differential DC voltage, which can result in uncompensated residual ISI. Histograms for the distribution of the simulated C-model input-referred offset are shown in Fig. 17 for the extreme boost settings. Notice that

the input-referred offset is higher for the high boost setting, as the mismatch in the tail current transistors has a larger negative impact than in the low boost case. In the next section, we will process the simulation measurements to make the A-model more representative of the measured C-model behavior.



(a) **(b)**

Fig. 17 – Measured CTLE input mismatch for (a) minimum and (b) maximum boost settings.

6.3 C-model driven A-model refresh

Once data from C-model characterization becomes available, the A-model can be updated to better reflect the true circuit behavior. The specification-based A-model, shown in Fig. 5, is output-swing-limited: the saturating amplifier is at the output of the CTLE filter. However, the DC analysis (Fig. 16(b)) of the C-model showed that the input swing limits linearity. Moreover, the input limitation is boost-dependent. These changes are reflected in the transposition of the CTLE filter and saturating amplifier in Fig. 18. Furthermore, the measured DC transfer curves (Fig. 16(b)) are specified as the input-to-output transfer curves to be used by the A-model saturating amplifier. Next, we describe the method used to update the CTLE A-model.

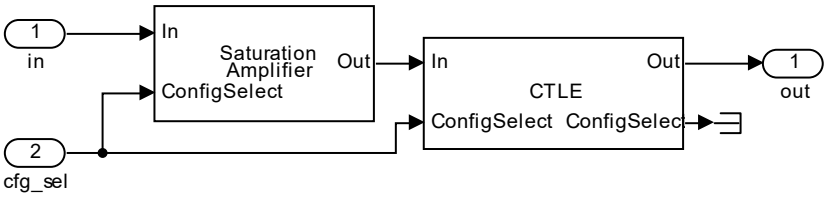


Fig. 18 – Circuit-simulation-refreshed CTLE model – note that the non-linearity is at the input and that the nonlinearity and CTLE are based on simulation measurements.

Updating the A-model of the CTLE frequency-dependent behavior based on the measured C-model behavior is done in two steps:

1. Determining a rational fit for the measured AC data, and
2. Updating the A-model CTLE to use the updated gain, pole, and zero values.

The family of measured C-model CTLE AC responses are approximated by rational functions to determine a set of poles and zeros for each CTLE boost setting [3-5]. Although several algorithms exist for rational approximation, we have found that CTLE responses are particularly well-suited to be fit by the AAA algorithm because this approach minimizes errors by a greedy selection of support points that avoid exponential instabilities [5]. The resulting rationally fitted gain-pole-zero (GPZ) matrix is used to configure the A-model CTLE.

We will now show that by leveraging just two standard analog simulation results, we have drastically increased the representativeness of the A-model. Different circuits may require different types of analog simulations; however, these simulations are typically performed as part of C-model validation activities. The complexity of the C-model characterization simulations can be increased for increased A-model fidelity. Furthermore, process corner support can also be added via a model parameter. Above all, the representativeness of the resulting A-model can be much better matched in MATLAB and Simulink than is possible via SystemVerilog language constructs. Next, we show that the transient A-model behavior correlates well with the C-model behavior.

6.4 Correlation between circuit and system model

The accuracy of the circuit-representative A-model can be evaluated using transient test signals that exercise both the A- and C-models. The captured C-model waveforms and the corresponding A-model waveforms for an input PRBS-7 sequence are shown in Fig. 19. The A- and C-models are exercised using 100- and 400-mV_P signal swings for the input PRBS waveform, and the CTLE is configured for minimum or maximum boost. Note that the A- and C-model output waveforms are very well-correlated. Discrepancy in the 400-mV_P high-boost A-model output is due to the CTLE being both input- and output-limited. Having shown that the A- and C-models correlate, we will now re-export an updated B-model.

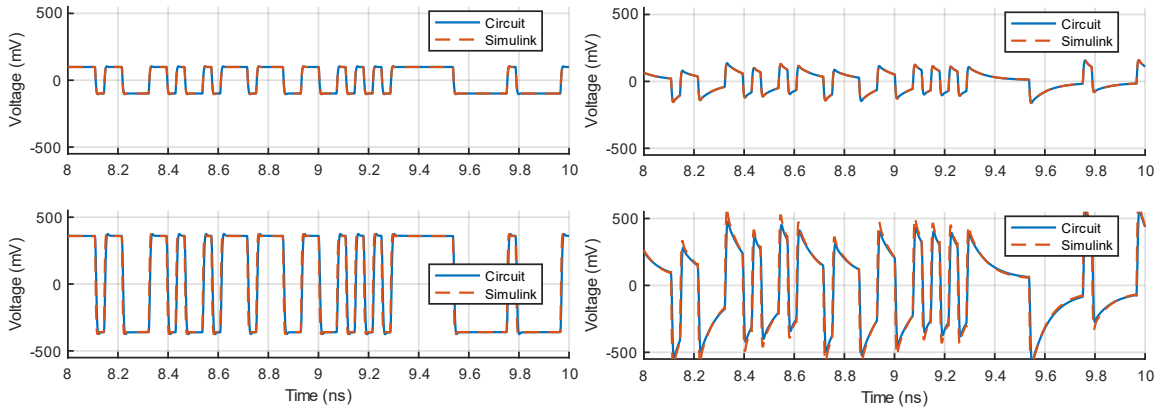


Fig. 19 – 100- and 400-mV_P circuit-simulated and Simulink model PRBS pattern response waveforms for (a) minimum and (b) maximum boost settings.

6.5 Automatic model refresh

The exact same flow used for exporting the COM-based CTLE model is used to update the B-model. No changes are needed to the CTLE export template nor the A-to-B-model interface gasket; the only action needed is to re-generate the DLL from Simulink using the updated A-model. The A-to-B-model transient correlation plots for the updated A- and B-models is shown in Fig. 20; there is no apparent difference between the two model outputs.

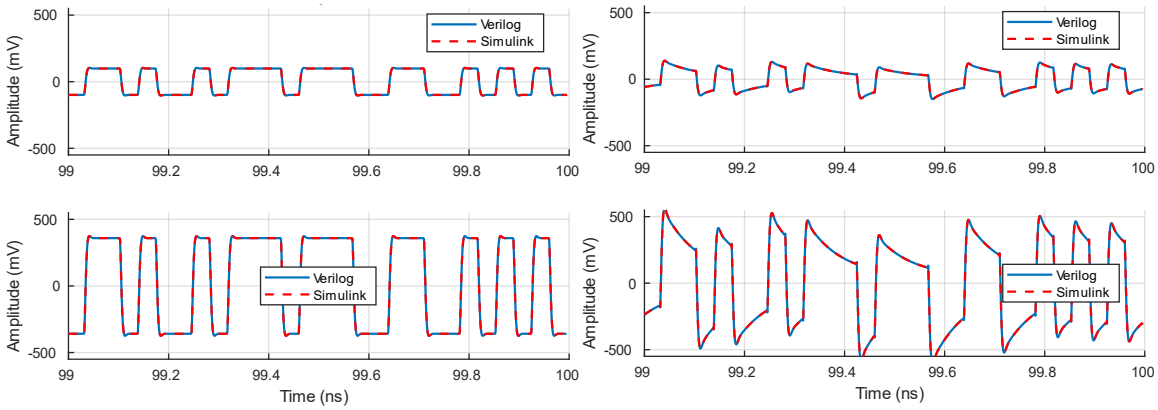


Fig. 20 – Verilog and Simulink output waveforms for circuit-representative CTLE driven by a PRBS waveform with a (top) 100-mV and (bottom) 400-mV amplitude for (a) minimum and (b) maximum boost settings.

7 Conclusion

This paper demonstrated a methodology for left-shifting SerDes mixed-signal validation, by leveraging early A-models along with defined C-model interfaces to generate B-models that can be used for analog blocks in mixed-signal simulations. As C-models mature and are characterized by analog simulations, the simulation data is used to update and re-align the A-models to reflect the C-model behavior more accurately. This flow is independent on C-model completion; in fact, the process of C-model recharacterization and A-model refresh can be repeated many times, and at each step an up-to-date B-model is automatically available for mixed-signal simulations. Unlike typical B-model generation approaches, which rely on human transcription of functionality or require completed C-models, the approach shown here does not require additional overhead: no overhead on top of the effort required for system-level trade-off analysis using A-models during a SerDes development cycle. The early access to B-models based on A-models allows for the left-shift of the SerDes validation effort.

References

[1] IEEEP802.ck Task Force 2020, “IEEE P802.3ck Task Force – Tools and Channels,” accessed February 14, 2022, <https://www.ieee802.org/3/ck/public/tools/index.html>

[2] IEEEP802.ck Task Force 2020, “IEEE P802.3ck Task Force – Ad Hoc Area,” accessed February 14, 2022, <https://grouper.ieee.org/groups/802/3/ck/public/adhoc/index.html>

[3] MathWorks, R2022a, “Ctlefitter,” The Mathworks, Inc., accessed February 14, 2022, <https://www.mathworks.com/help/serdes/ref/ctlefitter.html>

[4] Mathworks, R2021b, “Rational,” The Mathworks, Inc. accessed February 14, 2022, <https://www.mathworks.com/help/rf/ref/rfckt.rational.html>

[5] Y. Nakatsukasa, O. Sète, L.N. Trefethen, "The AAA algorithm for rational approximation", *SIAM Journal on Scientific Computing*, 2018

[6] MathWorks, R2021b, “Verification with UVM and SystemVerilog Components,” The Mathworks, Inc., accessed February 14, 2022, <https://www.mathworks.com/help/hdlverifier/systemverilog-dpi-generation.html>

[7] M. -A. LaCroix *et al*., "8.4 A 116Gb/s DSP-Based Wireline Transceiver in 7nm CMOS Achieving 6pJ/b at 45dB Loss in PAM-4/Duo-PAM-4 and 52dB in PAM-2," *IEEE International Solid- State Circuits Conference*, 2021, pp. 132-134

[8] D. Xu *et al*., "8.5 A Scalable Adaptive ADC/DSP-Based 1.25-to-56Gbps/112Gbps High-Speed Transceiver Architecture Using Decision-Directed MMSE CDR in 16nm and 7nm," *IEEE International Solid- State Circuits Conference*, 2021, pp. 134-136

[9] Lee, T. "ISSCC Special Event: Circuit Design and Testing Mistakes of Beginning Engineers." *IEEE Solid-State Circuits Magazine*, Spring 2021, pp. 111-118

[10] Chen, J. et al, “Mixed-Signal Methodology Guide,” Cadence Design System, Incorporated, 2012

[11] Carlson, S., “Top 5 Issues that Make Things Go Wrong in Mixed-Signal Verification,” accessed Feb 14, 2022, <https://community.cadence.com/cadence_blogs_8/b/ms/posts/top-5-issues-that-make-things-go-wrong-in-mixed-signal-verification>

[12] Lim, B. C., “Model Validation of Mixed-Signal Systems,” *PhD Dissertation*, Stanford University, 2012.

[13] Tarkiainen, J., “Mixed-Signal Verification of Analog IP using Schematic Model Generator and SystemVerilog,” *Master’s Thesis*, University of Oulu, April 2018.

[14] Bailey, B., “Increase in Analog Problems,” Semiconductor Engineering, Oct. 2020, accessed Feb 14, 2022, < https://semiengineering.com/increase-in-analog-problems/>