Design Methodologies and Automated Generation of Ultra High Speed Wireline SerDes Transmitters

Ayan Biswas

Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2023-213
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2023/EECS-2023-213.html

August 11, 2023

Acknowledgement

I wish to convey my utmost gratitude to my advisors, Prof Vladimir Stojanovic and Prof Elad Alon, and the other members of my committee, Prof Ali Niknejad and Prof Martin White.
I appreciate the efforts of my tape-out team: Paul Kwon, Kunmo Kim, Yi-Hsuan Shih, and Bob Zhou.
I thank the BWRC staff: Anita Flynn, Brian Richards for their support.
I thank all the funding agencies: DARPA, ComSenTer, Intel, AI4OPT, for the research grants that sponsored my projects.
I thank Eric Chang, Zhongkai Wang, Minsoo Choi, and others, for collaborating on various projects.
Last but not least, I thank my parents for always being my biggest pillars of strength, and motivating me to aim for nothing short of excellence in all my endeavors.

Design Methodologies and Automated Generation of Ultra High Speed Wireline SerDes Transmitters

By

Ayan Biswas

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Engineering — Electrical Engineering and Computer Sciences

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Vladimir Stojanović, Chair
Professor Elad Alon
Professor Ali Niknejad
Professor Martin White

Summer 2023

Design Methodologies and Automated Generation of Ultra High Speed Wireline SerDes Transmitters

Abstract

Design Methodologies and Automated Generation of Ultra High Speed Wireline SerDes Transmitters

by

Ayan Biswas

Doctor of Philosophy in Engineering — Electrical Engineering and Computer Sciences

University of California, Berkeley

Professor Vladimir Stojanović, Chair

With the ever increasing bandwidth demand in high performance computing, network and communications, machine learning applications, etc, wireline data transfer between multiple chips on the same package has been doubling in per-lane data rate every 3-4 years. The challenging design complexity of the analog and mixed signal front-end circuits necessitates the use of automated process portable layout generators and closed loop design scripts to reduce the turn-around time from circuit design to tape-outs in advanced FinFET technology nodes.

To that end, this thesis introduces new feature improvements in the Berkeley Analog Generator (BAG) 3++, which is an open source framework for automating process portable circuit generation and encoding closed loop design methodologies. Then the thesis presents the design of a 160 Gbps NRZ transmitter (TX) targeting low loss (∼3dB) ultra short reach channels for die-to-die data transfer, with 2 tap FFE for pre-equalization. Some major challenges in this effort are the design of a high speed 8:1 mux stage using inductor-based peaking topologies, and a novel method of creating the 1 UI delayed data stream for the FFE precursor using the available octature clock phases. The TX is taped out in Intel16 process as a part of a complete 160 Gbps NRZ transceiver design, and tested in loopback mode by serially transmitting data to the receiver over a 8.5 mm differential channel on package.

To my parents

Contents

List of Figures

List of Tables

Acknowledgments

The PhD endeavor is a long, strenuous, and ultimately rewarding experience, and I am eternally grateful to many people for their guidance and support through this journey.

I wish to first convey my utmost gratitude to my advisors, Prof Vladimir Stojanović and Prof Elad Alon. Elad gave me the golden opportunity to pursue my PhD at UC Berkeley as a part of his research group at the Berkeley Wireless Research Center (BWRC), and entrusted me with the responsibility of maintaining the Berkeley Analog Generator (BAG) framework, one of the largest open source collaborative projects in our lab. His focus on developing design methodologies for analog and mixed signal (AMS) circuits has helped to strengthen my foundation of core circuit design knowledge. Vladimir's hands-on guidance has always been immensely helpful, especially during the stressful tape-out periods, as he was always ready to brainstorm new circuit ideas, optimization algorithms, floorplanning tricks, etc. on a whiteboard whenever I got stuck in a seemingly infeasible design step.

I wish to thank the other members of my committee, Prof Ali Niknejad and Prof Martin White, for their helpful comments during my qualifying exam and dissertation talk. Ali provided extremely valuable guidance for designing the challenging inductive peaking topologies for bandwidth extension of the high speed stages in my transmitter datapath design.

Tape-outs can be one of the most difficult experiences of a circuit designer's life, especially in a PhD setting because of the small team size. That's why I have to give a huge shout-out to my tape-out team: Paul Kwon, Kunmo Kim, Yi-Hsuan Shih, and Bob Zhou, for making the tape-out experiences much more enjoyable by their dedicated work ethics, and immense help with debugging miscellaneous issues.

The BWRC staff members are often the unsung heroes, working behind the scenes to keep the lab and servers running, and helping students with guidance on different problems. I especially thank Anita Flynn for helping me learn the various steps of PCB design, and also teaching me how to communicate efficiently with external vendors. Brian Richards helped me resolve licensing issues of various CAD tools and technology nodes, that I ran into as a part of my BAG maintenance efforts.

I thank all the funding agencies: DARPA, ComSenTer, Intel, AI4OPT, for the research grants that sponsored my projects. Intel's University Shuttle Program gave me the opportunity to use Intel's advanced FinFET nodes for my research work.

Special mention needs to be made of Eric Chang for being an amazing mentor during my initial PhD years at BWRC, and then during my 2 years of internship at Blue Cheetah Analog Design, Inc. As the previous lead developer and maintainer of BAG before me, he trained me to be a perfectionist, and to cultivate software and hardware debugging skills.

Being a part of BWRC comes with the amazing perk of working with some of the brightest minds in my field of research, and I thank Zhongkai Wang, Minsoo Choi, Sean Huang, Zhaokai Liu, and many other incredible students in my cohort, for collaborating on various research projects and class projects over the years.

Last but not least, I thank my parents for always being my biggest pillars of strength, and motivating me to aim for nothing short of excellence in all my endeavors.

Chapter 1

Berkeley Analog Generator (BAG) 3++

1.1 Overview and Development History

Since the invention of the first monolithic integrated circuit chip by Robert Noyce at Fairchild Semiconductor in 1959, the design of integrated circuits (ICs) has taken multiple quantum leaps to span a multi-billion dollar industry today. The number of transistors on a chip went from tens to billions, while the technology node advanced from µm to nm scale to accommodate multi-functional multi-core processors in single chips. This significantly increases the design and layout complexity, and number of necessary engineer-hours to successfully tape-out a new design.

University of California, Berkeley (UCB) has been at the forefront of integrated circuit design efforts over the years, including the development of SPICE (Simulated Program with Integrated Circuit Emphasis) by Laurence Nagel and Prof Donald Pederson at the Electronics Research Laboratory in 1973. All modern analog and mixed signal (AMS) simulation tools (Spectre, NgSpice, HSpice, etc) have their roots in the SPICE project.

On the layout front, engineers advanced from hand drawing layouts using opaque tapes and films to using IC layout editor software. Currently, the most widely used industry standard EDA (electronic design automation) tool, specifically for analog and mixed signal designs, is Cadence's Virtuoso Design Suite that provides layout and schematic editors, simulation environment and waveform viewers, and integration with various other verification tools. In the Virtuoso Layout Editor, one can draw polygons and instantiate custom pcells (programmable cells) manually to complete the top level layout, or even use the proprietary SKILL language to program parts of the layout process.

While digital IC design has been streamlined over the years to take Verilog code as input and create complete layouts using automatic placement and routing (PnR) algorithms to optimize the timing margins on critical design paths, creating automatic PnR tools for analog and mixed signal designs still remains a challenge because of the additional com-

plexities like matched / common centroid transistor layout, passive elements like inductors, resistors, capacitors, etc. Human layout engineers learn these critical layout design tricks through years of experience, but it is difficult to encode all these rules into an automatic PnR program and get comparable performance to a human created design. Recent advances in AI (artificial intelligence) have resulted in some promising initial efforts (ALIGN (Analog Layout, Intelligently Generated from Netlists, MAGICAL (Machine Generated Analog IC Layout), etc.), but a lot of ground still remains to be covered.

At the Berkeley Wireless Research Center (BWRC) at UCB, researchers took a different approach to this problem, focusing on automated design generation rather than automatic generation. The core idea is to enable AMS designers to encode the layout strategy and design methodology in Python into the Berkeley Analog Generator (BAG) framework. The first publicly released version was BAG 2 [1], developed by Eric Chang, Jaeduk Han and others, and its versatility was demonstrated through the design of high performance ADCs and SerDes transceivers.

1.2 Core principle of BAG

In traditional AMS design flow, engineers share frozen IP (intellectual property) instances with other design teams. These instances are provided as black boxes and only expose the I/O (input and output) terminals while hiding the internal implementation details. These also usually come with fixed form factors and locations of the I/O terminals that cannot be modified to optimize the performance of the top level design that is integrating the IP instance, which can often limit the re-usability. The major motivation of BAG is to allow the re-use of the circuit design methodology rather than a frozen IP instance. An instance in BAG ideally comes with parametrized layout and schematic generators, and a top level design script that can take target specifications as input, run necessary simulations, post process the simulation results, and do some iterative optimization to produce a final design instance that satisfies the goal. It is important to note that none of these steps are 'automatic', instead the designer has complete control over every single step in the process, from setting up the required measurement procedures, to defining the location of every single device and wire used in the layout. This also allows the design to be process-portable, i.e. the circuit can be re-designed in a new technology node or process development kit (PDK) since the core design method remains the same. Hence, the main advantage of using BAG over a traditional design flow is being able to leverage computer resources to speed up the iterative design and verification process. BAG interfaces heavily with Virtuoso, making it easier to adopt for traditional AMS designers.

As shown in Fig 1.1, the target performance specifications and the design engineer's 'brain' (i.e. the design methodology) is encoded in the circuit generator (which covers layout and schematic generators as well as measurement and design scripts). The BAG framework takes this circuit generator(s), reads in the PDK collateral technology files provided by the foundry (including simulation models of all available devices, pcell and standard cell libraries,

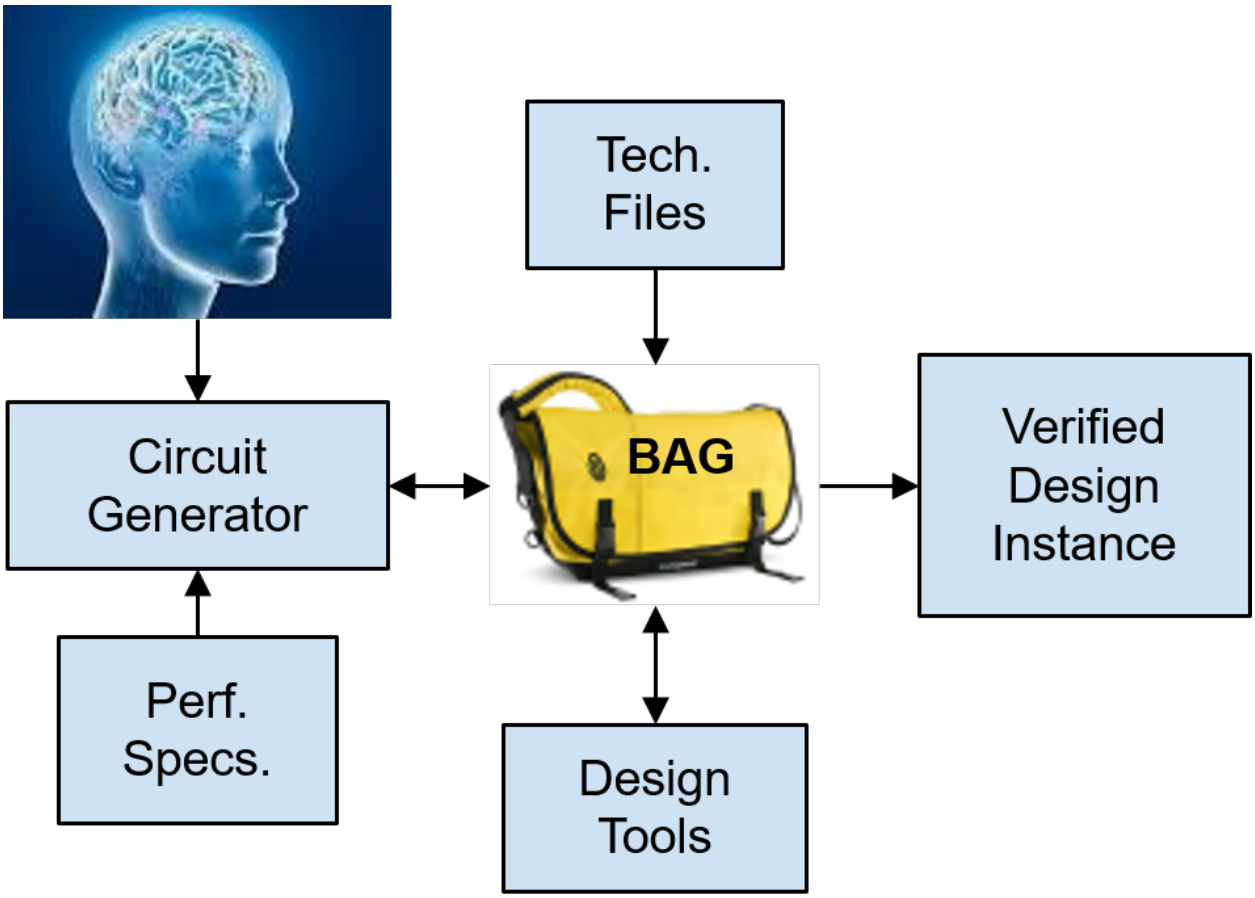


Figure 1.1: Overview of Berkeley Analog Generator (BAG) framework

etc), interfaces with industry standard and / or open source design tools (e.g. Cadence Virtuoso Design Suite, parasitic extraction tools like StarRC, QRC, etc, electromagnetic simulation tools like EMX, verification tools like Calibre, ICV, PVS, etc) and ideally outputs a verified design instance ready for tapeout or for integration in a higher level of hierarchy.

1.3 Improvements in BAG 3.0 and BAG 3++

While the core principle of the BAG framework remains the same, the implementation details have changed over the years based on feedback from multiple engineers to alleviate various issues with the API (application programming interface). One major update in transitioning to BAG 3.0 from BAG 2 was moving the back-end from Python to C++ to speed-up various layout geometry manipulation and netlisting calls. BAG 3.0 was developed primarily by Eric Chang at BWRC and at Blue Cheetah Analog Design Inc. (BCA). Subsequently, at BWRC, I led a research team to develop and open source BAG 3++ with more fea-

ture improvements and bug fixes. The current documentation for BAG 3++ is available at `https://bag3-readthedocs.readthedocs.io/en/latest/`, but it is still under active development as of August 2023.

The major changes in BAG 3.0 and BAG 3++ over BAG 2 are summarized in the following subsections:

Updates for Generators

Layout and schematic generators have undergone significant updates, with the back-end being entirely shifted to C++ that allowed many new exciting features to be implemented while simultaneously enabling faster run times. One major speed bottleneck in BAG 2 was interfacing to Virtuoso using the SKILL interface. This is now resolved by various workarounds mentioned later.

Some of the features are highlighted below.

1. Device-under-test (DUT) generators

    a) Layout generators:

    - Generated layouts are more compact than BAG 2, by allowing the designer to explicitly define the location of substrate rows and columns instead of automatically placing extra redundant taps for every subcell which would increase the area overhead.
    - Special generalized routing routines (e.g. `connect_via_stack()`, multiple resistor routing methods, etc) are available to ease the routing methods in layout generators and remove lots of boiler-plate code.
    - Partial layouts can be generated from the top level layout. This feature is detailed more in future chapters, where it's used to create subcells for electromagnetic simulations, and also to speed up circuit simulation steps by selectively omitting certain blocks.
    - BAG exports the generated layout directly to the GDS format, which can then be streamed in to Virtuoso. This is faster than SKILL calls to draw the layout in Virtuoso, since the layout geometry computation is handled in Python and C++.
    - Manual layouts can be exported as GDS and then dropped into the BAG layouts as blackboxes. This allows users to place any manually edited subcell layout, or some standard cell layouts provided by the PDK, directly in the top level layout. Users need to provide information about the overall size and location of all I/O terminals of the blackboxed cell to BAG, as well as any top level routing wires for collision checking at upper levels of hierarchy.

    b) Schematic generators: BAG stores an internal representation of the entire circuit hierarchy tree, and can emit various netlist formats from it. For the Virtuoso

representation, it still relies on SKILL calls to manipulate template schematics into final generated schematics. Hence this step is slow and only recommended for debugging purposes, while the much faster direct netlist generation feature is used for closed loop design and measurement scripts.

c) Behavioral model generators: Using the internal representation of the circuit hierarchy, BAG can also emit behavioral models using user provided models of the leaf cells. In a traditional AMS flow, users write a top level behavioral model of the entire system, and may unintentionally miss certain key aspects of the circuits or miss some last minute schematic changes during the tapeout process. Generating a hierarchical behavioral model using BAG ensures that the model always accurately represents the schematic. A key application of this feature is highlighted in Chapter 6.

2. Testbench (TB) generators:

a) `GenericTB`: All the generic testbenches available in Spectre are now available as Python classes in BAG (e.g. `DCTB`, `ACTB`, `TranTB`, `SPTB`, `PSSTB` etc. for DC, AC, transient, s parameter, periodic steady state analyses respectively, among others). Users are recommended to directly configure these existing classes instead of creating new custom testbenches, since these generic classes are fully parametrized and reconfigurable based on the application.

b) Harnesses: One major limitation of BAG 2 was that it used to allow one TB to have only one DUT in addition to various `analogLib` library components for input stimulus, loading, etc. But designers often require multiple DUTs in the same TB to determine the effect of different subcells interfacing with each other, before the top level has been completely assembled. BAG 3++ enables this by introducing the concept of harnesses. In BAG terminology, each TB still has one DUT, which is the main subcell being tested, but allows users to instantiate multiple harnesses, which are other subcells for driving or loading the main DUT.

c) `nport`: s-parameter files from electromagnetic (EM) simulations can be instantiated in the TB using the `nport` component from `analogLib` library. This enables the circuit simulation of inductors, channel models, etc.

d) VerilogA: BAG 3++ introduces VerilogA integration in testbenches. Any components from the `ahdlLib` library or any custom VerilogA module can now be directly instantiated in the TB, to serve as input stimulus, load, AMS model of any complex sub system, etc. as needed by the application. In particular, this allows the efficient encoding of many complex input stimuli that were previously provided using `pwlf` (piecewise linear function) files only.

Updates for Measurement API

The measurement API in BAG 2 used to be intimidatingly difficult to use, and slow because of the Virtuoso ADEXL / Maestro Interface. As a result, most users used only the automated layout and schematic generation features in BAG 2, while ignoring the measurement aspect completely and doing traditional simulation testbenches in Virtuoso. Hence, a major motivation for BAG 3.0 and BAG 3++ was the development of a more user friendly measurement API. Some major highlights are summarized below.

1. `SimAccess`: This is a Python class that takes the DUT and harness netlists created from BAG's internal circuit representation, natively writes a Spectre simulation netlist of the TB using the DUT and harness netlists, directly executes Spectre on the command line with user specified additional arguments, and formats the Spectre output data into a special `SimData` class in BAG for further post processing. As a result it completely avoids the ADEXL interface and is just limited by the Spectre runtime. It supports three different spectre output formats (psfxl, psfbin, nutbin) which have their individual pros and cons.

    a) psfxl: This is the default Spectre output format used in Virtuoso. It can be processed only using Cadence's proprietary Simulation Results Reader (SRR) program, and is the slowest among the three options. However, it covers every simulation setting available in Spectre.

    b) psfbin: This used to be the default Spectre output format in Virtuoso before psfxl was created. BAG uses the libpsf program to parse this. While this is faster than SRR, it has been found to error for certain transient simulations with large number of saved outputs. BAG 3++ is configured to use this option by default.

    c) nutbin: This is a SPICE output format, so it works for Spectre as well as other SPICE variants. This is parsed natively in BAG's `NutBinParser` class and is the fastest among the three options. However, the nutbin format doesn't support some advanced Spectre features like multithreading and multiprocessing, and multiple sidebands in PAC (periodic AC) simulations.

2. `EmSimAccess`: This is a Python class that can take a GDS file, run EMX on it and return the output s parameter file, along with optional subsequent post processing. BAG 3++ introduces this class for automated characterization of magnetic passives in a top level design script (as elaborated in Chapter 2) or for stand-alone cases.

3. `GatherHelper`: This class uses Python's `asyncio` feature to enable launching multiple asynchronous subprocesses in parallel (e.g. parallel simulation and measurements across different PVT variations, etc). It can also leverage LSF queues for better server utilization.

Design Scripts

Design scripts were another seldom used BAG 2 feature because of its intimidating measurement API. It's much more user friendly now and comes with a built-in configurable optimization routine.

1. `DesignerBase`: This is the base Python class for writing any design script. It can be used to generate multiple instances of the DUT with varying parameters, launch parallel measurement scripts on each of them, converge on a design based on target specifications, and run sign-off verification measurements of the final design. Usually design scripts fall in one of two different categories:

    - Iterative / serial design scripts: As illustrated in Fig 1.2, these start off with some initial design parameters derived analytically based on the target specifications and the device characteristics of the PDK. After post processing the measurement results of the DUT and comparing with the target, its parameters are tweaked based on some algorithm defined by the user to create a new DUT for re-measurements. This process continues to iterate until the measurement results meet the target. This type of design scripts is suitable mostly for designs with a single optimization variable.

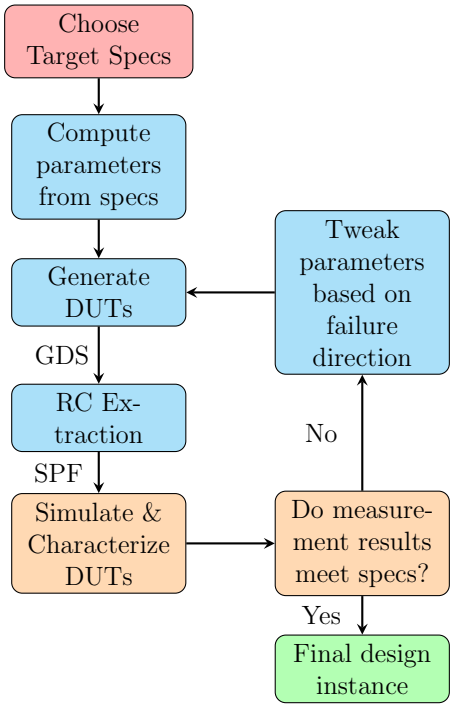


Figure 1.2: Flowchart for Iterative / Serial Design Script

- Parallel design scripts: For designs with multiple optimization variables, it is usually very difficult to analytically determine the best next step in the iterative design script process. Hence parallel design scripts follow a different strategy by allowing the user to define a constrained search space. Multiple DUTs are created by drawing parameters from the search space, and then the measurement results are compared to find the optimal DUT based on the target specifications.

2. `OptDesigner`: This is a special subclass of DesignerBase, created by my collaborator Paul Kwon, that implements parallel design scripts explained above. As illustrated in Fig 1.3, it first iterates over the user specified design space to create a large characterization database, by sweeping the design and simulation variables, and returns a multi-dimensional interpolator across the measurement results. Then it uses the target specifications and constraints provided by the user and optimizes across the interpolator using `scipy.optimize()` to converge to the optimal design.

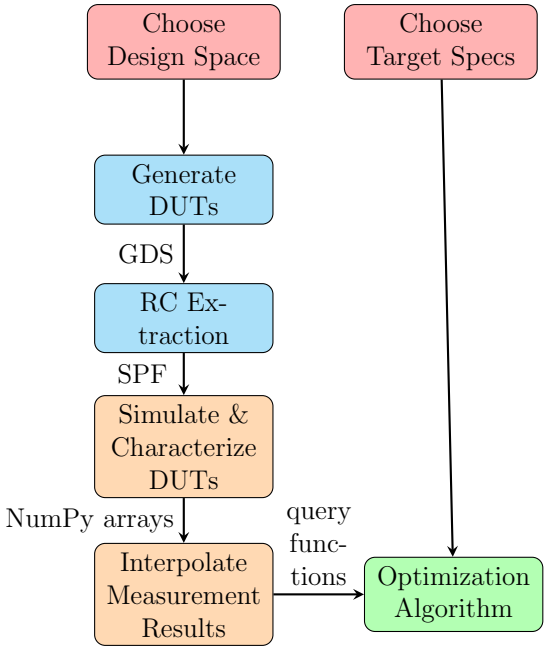


Figure 1.3: *OptDesigner* co-optimization loop

An advanced version of `OptDesigner` that also handles electromagnetic simulation of magnetic passives, by exploiting the `EmSimAccess` and partial layout generation features of BAG 3++, is elaborated in Chapter 2.

Conclusion

BAG 3++ provides substantial improvements over BAG 2 in terms of usability of the Python API, more efficient generated circuit layouts, and extensive measurement and design scripting abilities, while still maintaining a close integration with Virtuoso Design Suite for viewing layouts, schematics, and waveforms (using ViVa waveform browser). Additionally the generator and measurement APIs are completely stand-alone components, hence users can run the measurement scripts on manually created designs, or generate designs and manually create traditional Virtuoso testbenches. This provides extreme versatility as new users can ramp up on using specific sub features of BAG 3++ without learning all the details from the get-go. To the best of my knowledge, among all other newer competitor generator programs available today, BAG 3++ stands out as the most versatile automated generator framework with the broadest coverage of AMS design flow, and integration with industry standard EDA tools.

As part of my PhD research, I have developed and maintained BAG 3++ layout primitives across seven widely different PDKs spanning multiple technologies (planar, SOI, FinFETs) from multiple foundries (TSMC, Global Foundries, Intel, SkyWater). The lessons learned from writing generators that are process portable across such a wide PDK variation helped me to not only enable new features and improve the BAG 3++ framework, but also to optimize the PDK specific implementations and understand how to push the circuit performance across many different constraints and challenges.

Chapter 2

Automated Process-portable Electromagnetic and Circuit Co-optimization using BAG 3++

2.1 Problem Setup

Electromagnetic (EM) simulations and optimization of magnetic passives like inductors, t-coils, transmission lines, etc. is one of the most challenging aspects of AMS design. Even though simple theoretical models can be used to analytically study the first order effect of these magnetic passive circuit components, they do not capture all the higher order non linear effects of parasitics that can critically affect the performance. A few reasons that make EM optimization challenging are:

1. With technology scaling down to nm and sub nm processes, active device sizing has shrunk down to the extent where parasitic resistances and capacitances have started to become major contributors to the performance of circuits. For inductors or t-coils with multiple turns or across multiple metal layers, side wall capacitance between the turns or capacitance across metal layers can lower the self resonant frequency of the inductor or t-coil, thereby reducing the benefits of using those as peaking structures for high speed applications.

2. For ultra high speed wireline or wireless applications, the wavelength can become comparable to the length of long routing wires, so they have to be treated as transmission lines, and require adequate shielding and isolation to maintain the line impedance $Z_o = \sqrt{\frac{L}{C}}$, where L is the inductance of the transmission line and C is the parasitic capacitance of the transmission line to the substrate or neighboring wires. This introduces extra complexity in the design process since these effects are not captured in schematic simulations, and have to be handled carefully in parasitic and EM extracted simulations.

3. For narrow band applications like LC oscillators, a simple linear model of the inductor can be useful for theoretical analysis. This is because the behavior of the inductor can be curve-fit to a linear model with lumped components across a narrow frequency band around the center frequency of interest. Integrand's ModelGen tool (packaged with EMX) can create these 'simple' lumped models of various magnetic passives using `analogLib` passive components. However, these models are not suitable for wide band applications like extending amplifier bandwidth using peaking structures, since a lumped circuit model cannot adequately represent the more complex higher order behavior of the inductor or t-coil all the way from DC to 10's or 100's of GHz frequencies. So the only way of optimizing the design for wide band applications is to directly use the s parameters from EM simulations of the magnetic passives, and adjust the layout geometry based on knowledge gained from experience and trial-and-error.

The unavailability of simple analytical models of the magnetic passives, and reliance on trial-and-error to perform EM optimization, makes this a perfect candidate for demonstrating the power of design scripts in BAG 3++, particularly the `OptDesigner` class described in Chapter 1. Designers can define a constrained search space for the active device parameters and also the magnetic passives parameters (like width of turns, spacing between turns, shape of the inductor or t-coil, radius in X and Y directions) based on floorplanning restrictions, PDK design rules, first-pass analysis, and prior experience. Then the `OptDesigner` class will create multiple DUTs spanning all the parameter combinations from the user defined search space, and simulate and measure each of them to create a large characterization database. The database stores the results in the form of `NumPy` arrays and can be interpolated to predict the performance of other intermediate parameter combinations in the search space that were not explicitly characterized. Then the optimization algorithm can traverse through the interpolated results to pick a DUT that satisfies the target specifications.

The characterization step is the most time consuming and can take on the order of hours to days depending upon the complexity of the design and measurements, and the size of the constrained search space. However, since the entire process is fully configurable and automated in BAG 3++, users can just launch the characterization in the background, thereby reducing the number of active engineer-hours needed for the design process. Once the characterization is completed, the database is cached, and can be queried instantaneously for the subsequent optimization step. Hence, if the search space does not change, multiple optimization iterations can be performed for different target specifications by just querying the cached database to get the optimal DUT and then running a final set of sign-off verification measurements on it. This significantly increases the designers' productivity by leveraging computer resources to handle the computational and time expensive trial-and-error steps. Since both the active device sizing as well as the magnetic passives sizing can be swept in the search space, this allows simultaneous EM and circuit co-optimization.

The BAG 2 framework was previously used to design inductors by querying an EM characterization database to achieve the desired inductor quality factor for an LC oscillator application [12]. However, this optimized the inductor in isolation from the rest of the cir-

cuit, thereby ignoring the effect of the routing parasitics from the inductor leads to the active devices. BAG 3++ drastically improves on this by doing EM and circuit co-optimization, in a more user friendly, readable and reusable setup, that makes it more suitable for wide-band applications. Prior work [4] has also used the EM derived s-parameters to directly optimize the desired circuit performance by iterating on the dimensions of the on-chip inductive peaking structures, signifying the relevance of this problem. BAG 3++ provides a publicly available open source process-portable framework to perform this EM and circuit co-optimization by interfacing with industry standard EDA tools.

This chapter demonstrates the co-optimization process in BAG 3++ by designing an output driver stage topology for 130 Gbps non-return-to-zero (NRZ) signalling. The process portability is demonstrated by running the optimizer in two drastically different technology nodes: Intel's 16 nm FinFET process and Global Foundries' 45 nm SPCLO process.

2.2 Advanced OptDesigner with Electromagnetic Simulations

The `OptDesigner` class illustrated in Fig 1.3 can only perform circuit optimization using Spectre simulation of schematic or parasitic extracted netlists. To enable electromagnetic simulation of magnetic passives, it was modified to the new advanced version in Fig 2.1.

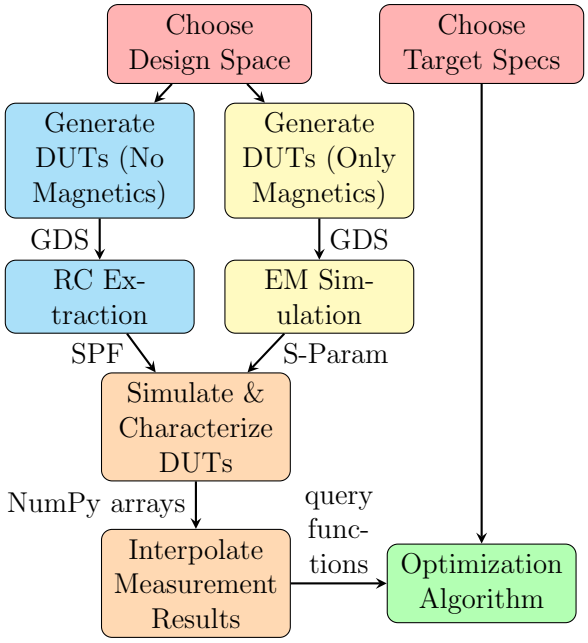


Figure 2.1: *OptDesigner* co-optimization loop

There is a new parallel path (in yellow) to the non-magnetic DUT generation and parasitic extraction path (in blue). This new path deals with the generation and EM simulation of the magnetic passives. The core operation of `OptDesigner` remains identical, but it can now use both the extracted parasitic netlists and the s-parameter files from the EM simulation for the Spectre simulation step.

2.3 Ultra High Speed Output Driver

Circuit Topology and Testbench

The schematic of the DUT and TB is shown in Fig 2.2.

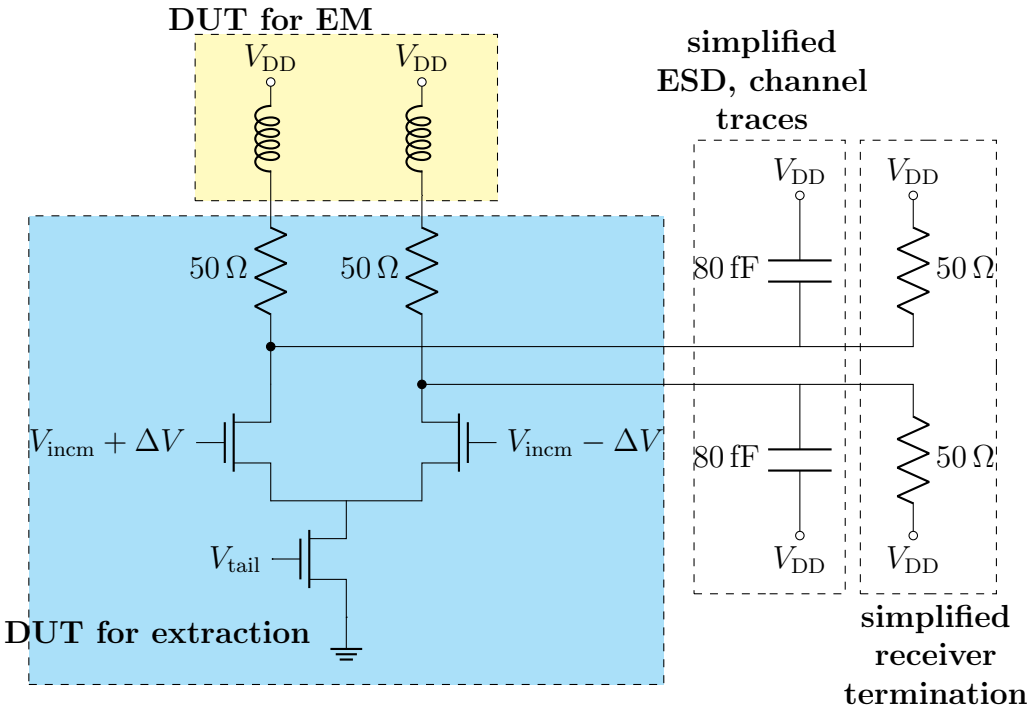


Figure 2.2: Testbench schematic, highlighting the partial DUTs used for RC extraction and EM simulation in Fig 2.1

The driver is an nmos differential pair with a 50 Ω resistor load and shunt peaking inductors. The resistors isolate the parasitic capacitance of the shunt peaking inductors from the high frequency output node, thereby removing them from the total output capacitance that has to be equalized by shunt peaking. The load is represented by the 80 fF capacitors which are a simplified lumped model of ESD and channel traces for the purpose of this demonstration. In a real design scenario, this can be replaced by the real channel s parameter model for

more accurate simulation results. The transistors and resistors of the output driver (boxed in blue) form the sub DUT for the parasitic extraction step, while the inductors (boxed in yellow) form the sub DUT for the EM simulation step in Fig 2.1.

Partial Layout Generation in BAG 3++

One crucial aspect for creating the sub DUTs for the two parallel paths in `OptDesigner` is the partial layout generation capability of BAG 3++. As mentioned earlier, if the inductors are generated in isolation, the effect of the inductor lead routing to the rest of the circuit will not be captured in the simulations. In a high speed application (130 Gbps NRZ with Nyquist frequency of 65 GHz), the parasitic effects of the routing can significantly impact the output driver performance, so it is crucial to account for that effect.

BAG 3++ allows users to generate the partial layouts by omitting some components while still preserving the top level routing wires, power grid, etc. Hence the partial layout of the shunt peaking inductors generated by BAG 3++ includes the lead routing of the inductors to the non-magnetic components of the circuit along with any top level power grid. This ensures that their EM effects are included in the s parameters which will be used in the subsequent Spectre simulation step.

Fig 2.3 shows the overall layout of the DUT from Fig 2.2 in the Intel16 process. Fig 2.4 shows the two partial layouts of the sub DUTs that will be used in the two parallel paths in `OptDesigner`.

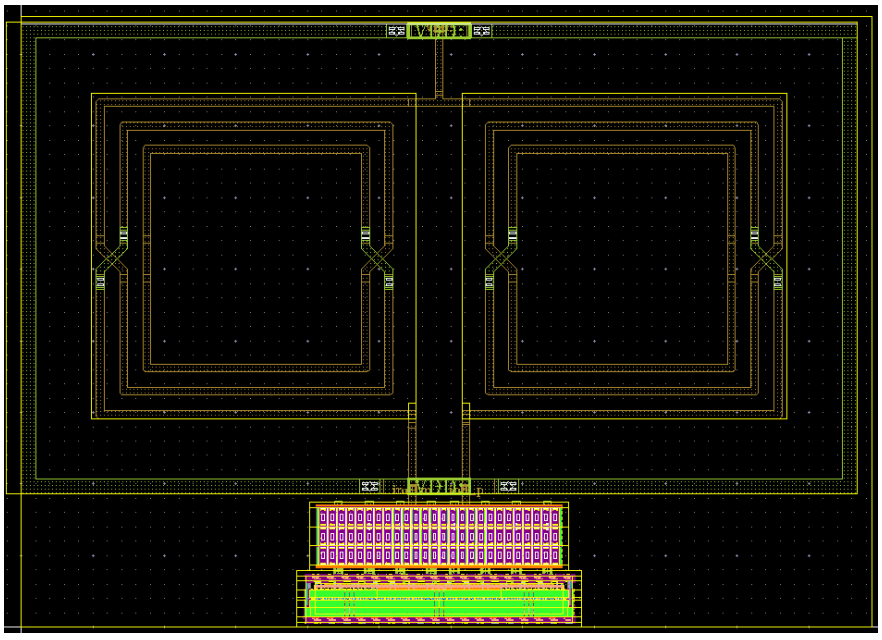
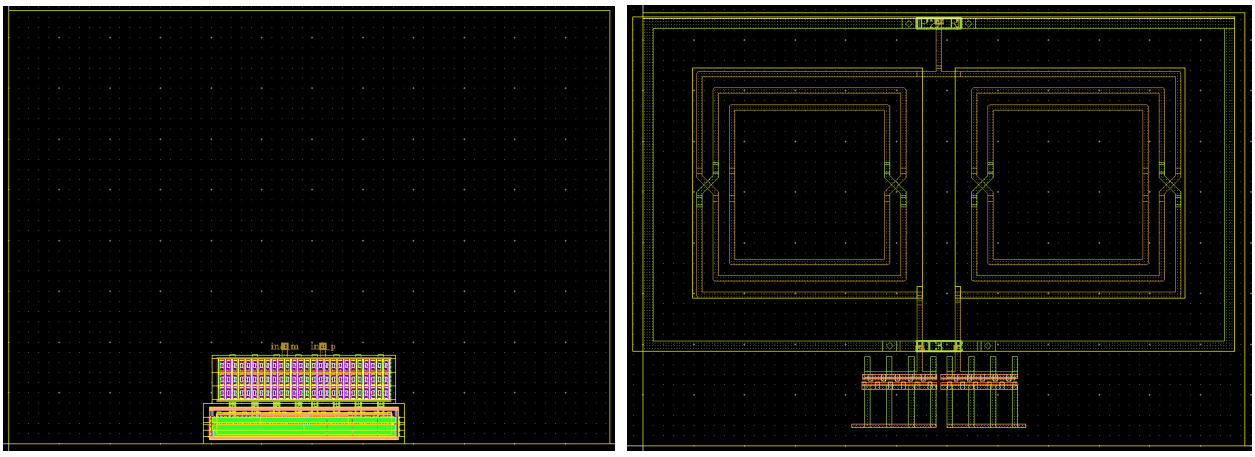


Figure 2.3: Full layout of the Output Driver DUT from Fig 2.2



(a) sub DUT for parasitic extraction (b) sub DUT for EM simulaion

Figure 2.4: Partial layouts of the Output Driver sub DUTs

As seen in the layout figure above, the sub DUT for EM simulation preserves the inductor lead routing to the non magnetic components, and other top level routing and power grid, to include their effects in the s parameters.

Measurement Setup and Design script

The `MeasurementManager` class in BAG 3++ is the base measurement class in Python that can be used to configure any of the available `GenericTB` testbench subclasses. As illustrated in the schematic in Fig 2.2, the output driver testbench sets up a transient noise simulation using a VerilogA random input bit stream module with differential $0.8\,\mathrm{V_{pp}}$. The Spectre simulation results are then analyzed using the `EyeAnalysis` class, which is a native post processing class in BAG 3++ that can create eye diagrams from transient simulation results and report the eye width and height.

The configuration settings of the `OptDesigner` class for this output driver are shown in Table 2.1. It should be noted that there is a distinction between design variables and simulation variables. Design variables refer to layout parameters that cannot be changed after tape-out, hence these have to be optimized across all PVT variations. Simulation variables refer to bias points that can be set using on-chip or off-chip calibration options like DACs (digital-to-analog converters), hence these can be separately optimized for each PVT corner.

For the purpose of this demonstration, the design variables are the number of fingers of the differential pair (seg_gm) and the tail transistor (seg_tail) on the blue path, and the radius of the inductor for the yellow path of Fig 2.1, while all other layout parameters are kept constant at some specific values to prune the search space. The simulation variables are the bias points for the input common mode of the differential pair (V_{incm}), and the gate of the tail

Table 2.1: *OptDesigner* Input Specifications

Parameters		**Sweep / Target values**
Corners		TT@25 °C, SS@−40 °C, FF@125 °C
Design variables	seg_gm	56, 64, 72, 80
	seg_tail	56, 64, 72, 80
	radius (resolution units)	18000, 22000, 26000, 30000
Simulation variables	V_{incm} (V)	0.65, 0.7, 0.75, 0.8
	V_{tail} (V)	0.65, 0.7, 0.75, 0.8
Speed		7.7 ps UI (130 Gbps)
Goal		Maximize eye height
Constraints	Eye width	$\geq$ 5.7 ps (0.75 UI)

transistor (V_{tail}). Since there are 4 values being swept for each of the variables, this implies that there will be $4^3 = 64$ design points characterized at $4^2 = 16$ simulation settings across 3 PVT corners, resulting in a total of 3072 simulations to create the full characterization database. As mentioned before, the characterization database stores results as `NumPy` arrays which can then be interpolated to predict the performance of other intermediate design and simulation points in the span of the search space.

2.4 Results

The `OptDesigner` class converges to fixed design parameters and corner-specific simulation parameters, as summarized in Table 2.2. The full design space characterization has a 7 hour runtime for each technology node, using distributed parallel processing on the BWRC server system. The characterization database is then cached, so that subsequent optimization runs for various target specifications by querying the database are instantaneous.

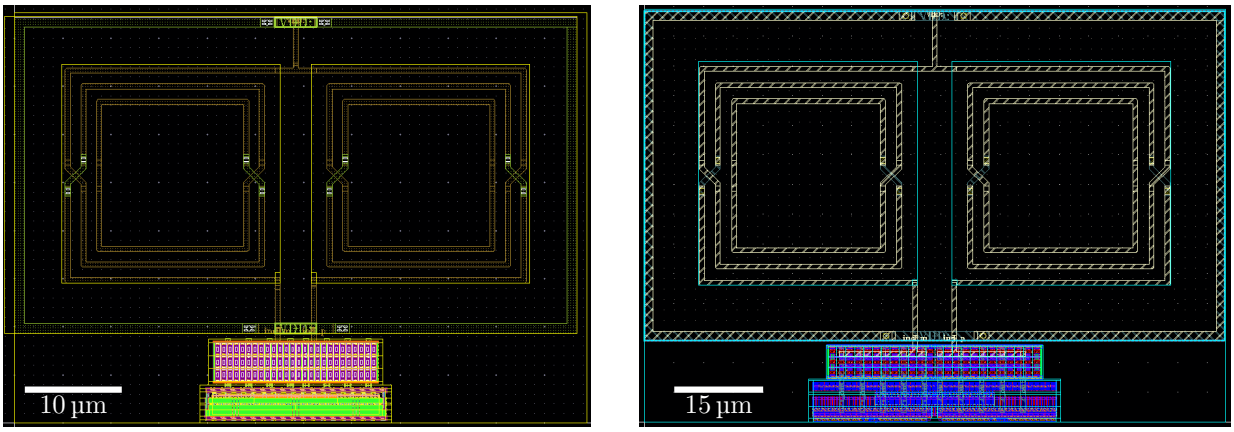
From the table, it can be seen that the design converged to using the largest allowed sizes of the transistors to maximize the eye height in both technology nodes, since there were no constraints specified for the power consumption. It picked a smaller inductor radius in the SOI process presumably because the layout parasitics are larger there, which reduce the self resonant frequency of larger sized inductors, thereby decreasing the equalization effect from the inductance. For the FinFET process, the optimal inductor radius size is not among the ones that were explicitly characterized in Table 2.1, showing the effect of interpolation across the `NumPy` arrays. The trend of the eye height across PVT variations is different in the two technology nodes because of PDK specific characteristics.

The complete layouts of the optimal DUTs in the two technology nodes based on the

Table 2.2: *OptDesigner* Results at $V_{DD} = 1\,V$

Technology node		16 nm **FinFET**			45 nm **SOI**		
Corners		TT,25	SS,-40	FF,125	TT,25	SS,-40	FF,125
Results	Eye height (V)	0.649	0.703	0.567	0.631	0.582	0.680
	Eye width (ps)	7.19	7.30	7.10	6.93	7.01	6.89
	Average current (mA)	18.3	19.5	18.5	21.1	17.0	26.5
Design variables	seg_gm	80			80		
	seg_tail	80			80		
	radius (resolution units)	21768			18000		
Simulation variables	V_{incm} (V)	0.681	0.744	0.681	0.771	0.771	0.800
	V_{tail} (V)	0.730	0.800	0.730	0.794	0.794	0.800

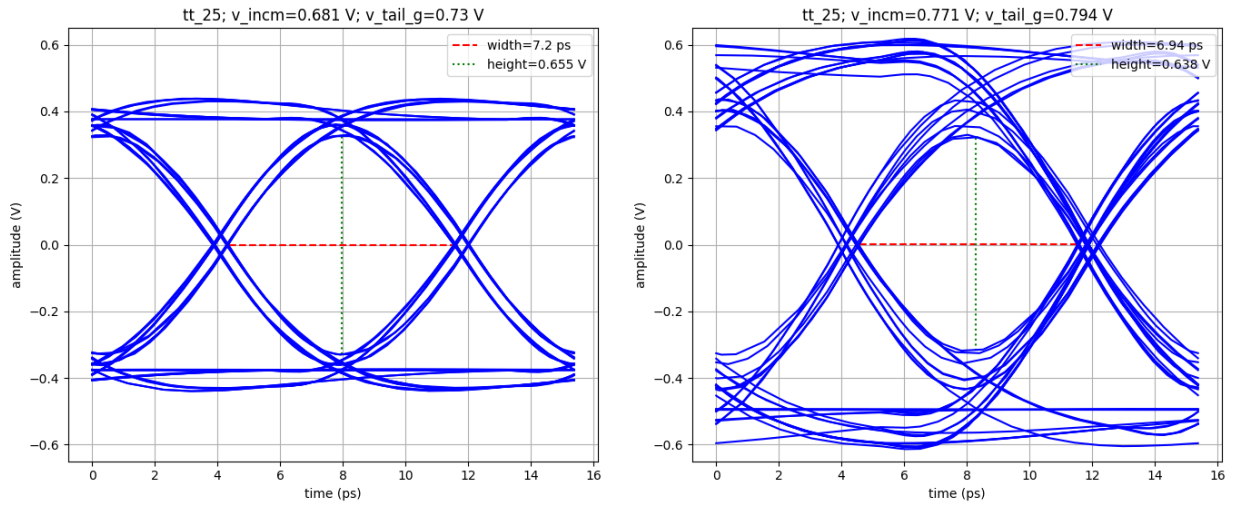
results in Table 2.2 are shown in Fig 2.5.



(a) 16 nm FinFET (b) 45 nm SOI

Figure 2.5: Optimal layouts based on Table 2.2

Fig 2.6 shows the corresponding eye diagrams for the TT@25 °C corner, using the bias conditions from Table 2.5. It can be seen that while the achieved eye heights are similar, the SOI process exhibits a more noisy eye, implying that the parasitics are larger and the performance is gain-bandwidth limited.



(a) 16 nm FinFET

(b) 45 nm SOI

Figure 2.6: Eye diagrams for TT@25 °C corner based on Table 2.2

Conclusion

It has been demonstrated that the fully automated EM and circuit co-optimization routine `OptDesigner` in BAG 3++ can use parametrized partial layout generation, followed by simultaneous EM simulation of magnetic structures and parasitic extraction of the remaining circuit, to design an ultra-high speed driver stage in two vastly different technology nodes. All the generators and scripts used in this chapter are open sourced and publicly available in the bag3_analog GitHub repository.

Chapter 3

SerDes Transmitter Datapath

3.1 Background

SerDes (**Ser**ializer - **Des**erializer) links originated from communication over fiber optic and coaxial links. Around the 1980's to late 1990's, it started being adopted for chip-to-chip (C2C) communication on PCBs and backplanes, and subsequently die-to-die (D2D) communication, replacing parallel links. The major advantage of SerDes over parallel links was that communication between two dies could now happen over one or two fibers instead of a parallel bus of fibers. Fig 3.1 shows a simplified diagram of a full SerDes link.

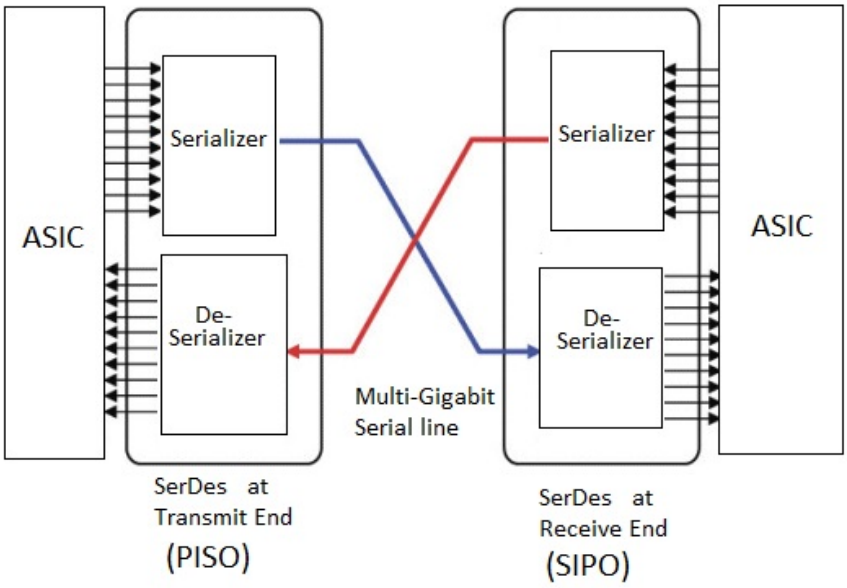


Figure 3.1: Overview of SerDes

In today's age, SerDes IP is ubiquitous in all SoCs (system on-chips) and data centers for high speed wireline I/O. Depending on the specific application, SerDes links can be classified into a few different categories based on the channel length (reach), the target data-rates and acceptable BERs (bit error rates). Fig 3.2 shows the different types of SerDes reaches.

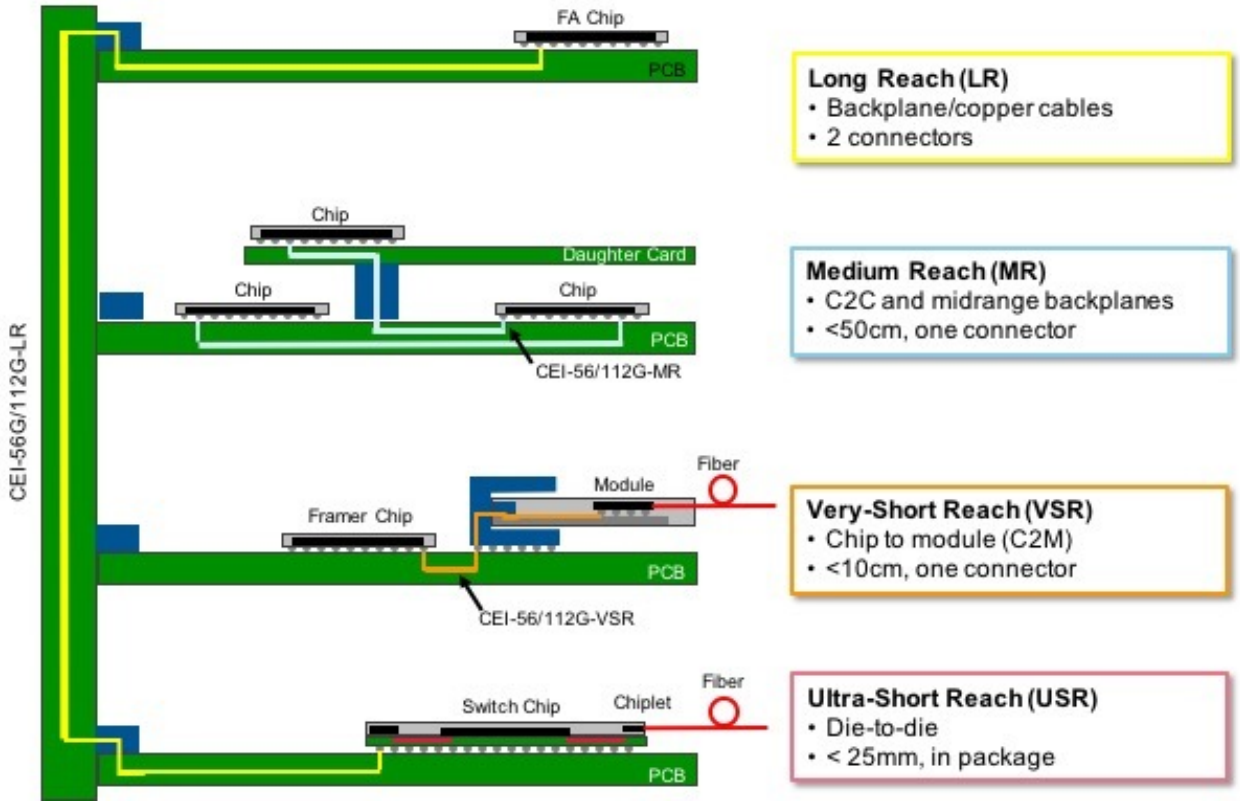
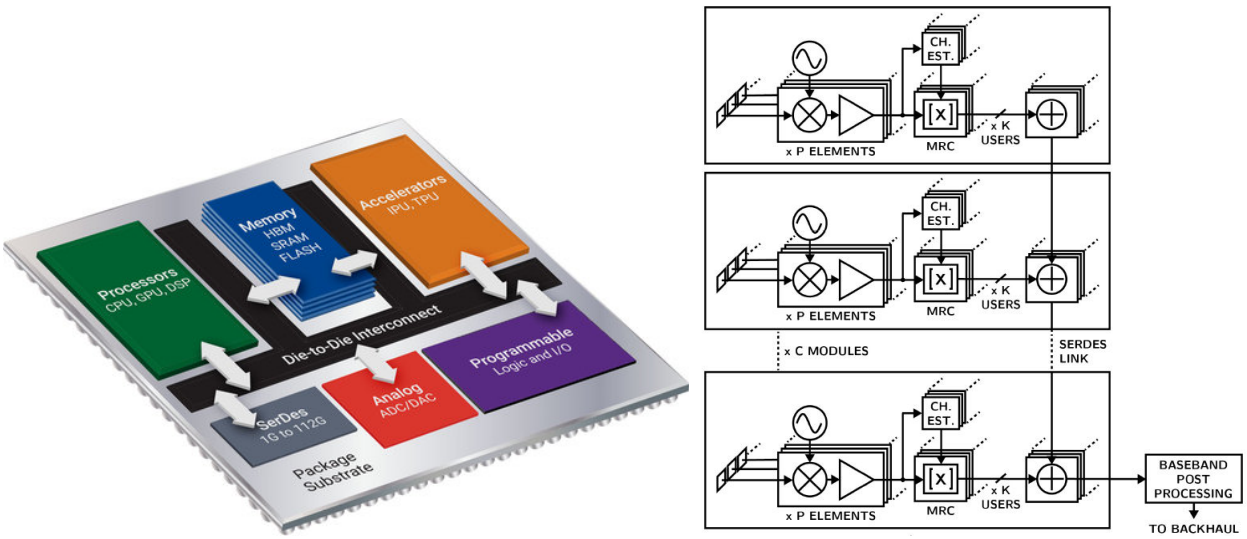


Figure 3.2: Different SerDes Reaches

Some common examples of SerDes applications are shown in Fig 3.3.

- Multi-chip packages (MCPs) use short reach SerDes for D2D interconnects and medium or long reach SerDes for C2C wireline data transfer.
- In massive MIMO arrays [6], appropriate reach SerDes is used to transfer data received by each user to the baseband processing units.

With the increasing bandwidth demand in high performance computing, networking and communications, machine learning applications, etc, industry trends show that wireline I/O has been doubling every 3-4 years, as illustrated in Fig 3.4 [3]. FinFET technology nodes have enabled SerDes transceivers to take advantage of process technology scaling and improve the data rate by gradually adopting digital - style architectures in the AMS components. Higher



(a) Multi chip package (b) Scalable Massive MIMO [6]

Figure 3.3: Common SerDes applications

cut-off frequencies, lower and more reliable parasitics make FinFETs much more attractive than mature planar technologies for ultra high speed applications.

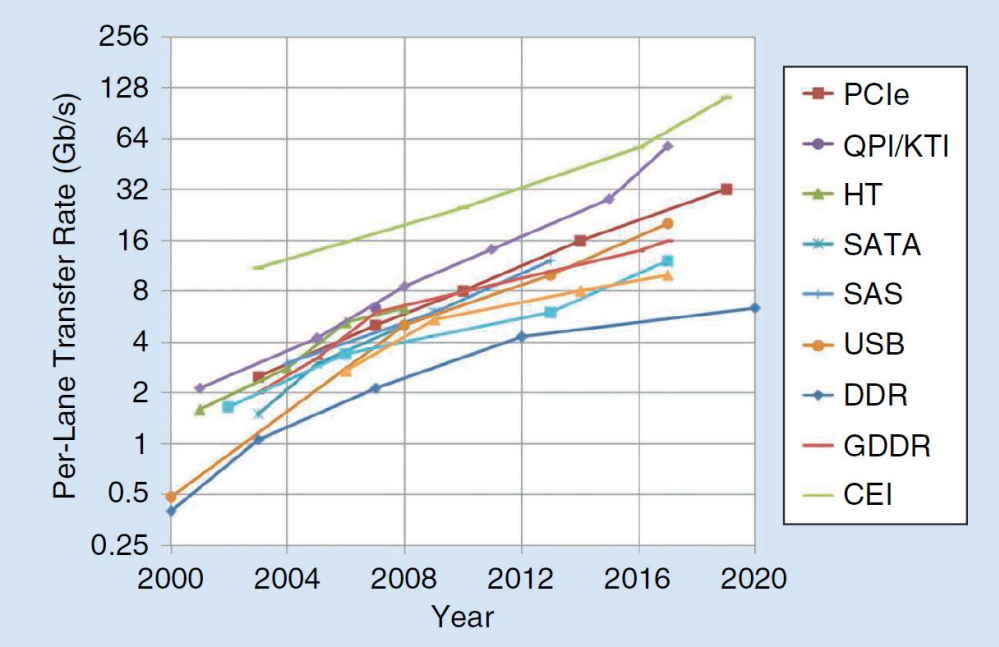


Figure 3.4: Wireline I/O trends [3]

Another approach to enable higher SerDes data-rates is using 4 level pulse-amplitude-modulation (PAM4) signalling rather than NRZ, as shown in Fig 3.5. For the same Nyquist frequency, PAM4 doubles the data-rate and provides better spectral efficiency. However, it causes a $\sim 9.5\,\text{dB}$ degradation in SNR (signal-to-noise ratio) since the vertical eye opening is only $\frac{1}{3}$ of a NRZ data eye, hence linearity and inter symbol interference (ISI) effects make the design of the equalization network extremely challenging.

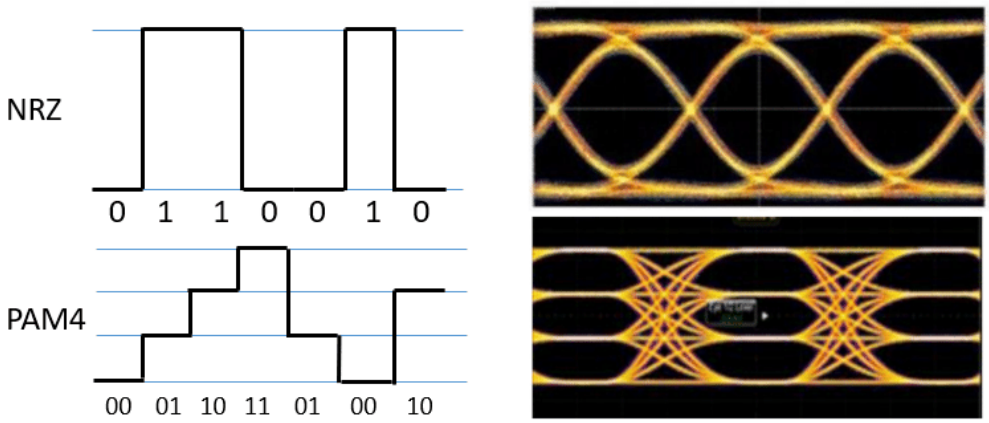


Figure 3.5: NRZ and PAM4 signalling waveforms and eye diagrams

Currently the state-of-the-art SerDes transmitters available in literature are a 224 Gbps PAM4, 112 Gbps NRZ transmitter with 8 tap FFE in 10 nm FinFET from Intel Labs [5] and a 200 Gbps PAM4, 100 Gbps NRZ transmitter with 5 tap FFE in 28 nm planar process from BWRC [13].

In recent years, there is a renewed interest in parallel links (e.g. UCIe, BoW) particularly for short reach applications. These allow higher shoreline bandwidth density because of densely packed bumps. Nevertheless, SerDes links still remain an exciting field of research. Additionally, the knowledge of the link budget analysis, circuit design methodologies, equalization techniques, etc are equally applicable in serial and parallel links design and optimization.

3.2 Motivation and Goals

FFE (feed forward equalizer) taps are used in SerDes transmitters (TX) and receivers (RX) for equalizing pre cursor ISI, while post cursor ISI is generally equalized using DFE (decision feedback equalizer) taps in the receiver. However, as data rates start to scale beyond 112 Gbps NRZ (56 GHz Nyquist frequency), it becomes increasingly difficult and near infeasible to close the DFE timing on the receiver. Hence alternative post cursor ISI cancellation techniques that rely on feed forward architectures only (i.e. no feedback) are being explored

in SerDes receiver research to alleviate the feedback-induced latency bottleneck. At BWRC, 1 tap feedforward MLSE (maximum likelihood sequence estimation) architectures are being explored for 160 Gbps NRZ signalling (80 GHz Nyquist frequency). Since the hardware implementation of the MLSE equalization scheme becomes significantly more complicated for PAM4 signalling, the focus is on just NRZ signalling in test chips for now.

Testing the receiver requires a corresponding 160 Gbps NRZ transmitter, which is the circuit design focus of this thesis. The target channel for the full transceiver system is an ultra short reach 8.5 mm differential channel on the organic substrate package, that is estimated to have $\sim$ 3 dB loss at the Nyquist frequency of 80 GHz. Based on link budget calculations, this channel can be equalized by 2 taps of FFE (1 main cursor and 1 pre cursor) on the transmitter and 1 tap MLSE based post cursor equalization on the receiver. This sets the top level target specifications of the transmitter as summarized in Table 3.1.

Table 3.1: Comparison of Current Goal with State-of-the-Art Transmitter Architectures

Specs	**JSSC 2022, Kim et al [5]**	**JSSC 2022, Wang et al [13]**	**Current Goal**
Technology node	10 nm FinFET	28 nm planar	16 nm FinFET
Data-rate (Gbps)	224 Gbps PAM4, 112 Gbps NRZ	200 Gbps PAM4, 100 Gbps NRZ	160 Gbps NRZ
Nyquist frequency (GHz)	56	50	80
Equalization technique	8 tap FFE	5 tap FFE	2 tap FFE

The target Nyquist frequency for this TX is higher than the current state-of-the-art, and there will be significant design challenges to achieve this goal. The estimated shoreline bandwidth density is comparable to state-of-the-art wireline I/O parallel links with standard packaging, as shown in Table 3.2.

The shoreline bandwidth density of the current work is estimated as follows:

- The C4 bump pitch is 245 µm.
- Assume that 4 transceivers can be packed at the die edge.
- Each transceiver requires GSSGSSG bumps, with the edge G shared with the adjacent transceiver.
- There is no bump overhead for clock signals since clock forwarding architecture is not used.

Table 3.2: Comparison of Shoreline Bandwidth Density

Package	**Architecture**	**Shoreline bandwidth density (Gbps / mm)**
Advanced	AIB	256 - 1024
	UCIe advanced	165 - 1317
Standard	UCIe standard	28 - 224
	BoW	100 - 1000+
	Current work	$\sim$ 200

- Total number of bump pitches for 4 transceivers is $6 \times 4 + 1 = 25$.
- Aggregate data-rate at the edge (in Gbps) is 160×2 (bi-directional) $\times 4$ (number of transceivers) $= 1280$.
- Hence shoreline bandwidth density is $\frac{1280 \text{ Gbps}}{25 \times 245\,\mu\text{m}} = \sim 209$ Gbps / mm

Theoretical Analysis using Process Parameters

This subsection studies the theoretical feasibility of the data-rate target based on the intrinsic gain-bandwidth (GBW) limit of the technology.

Consider an nmos input common source (CS) amplifier stage, as shown in Fig 3.6.

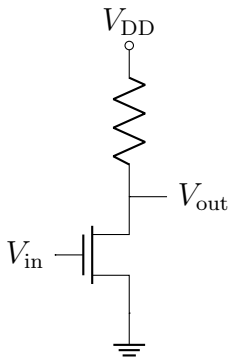


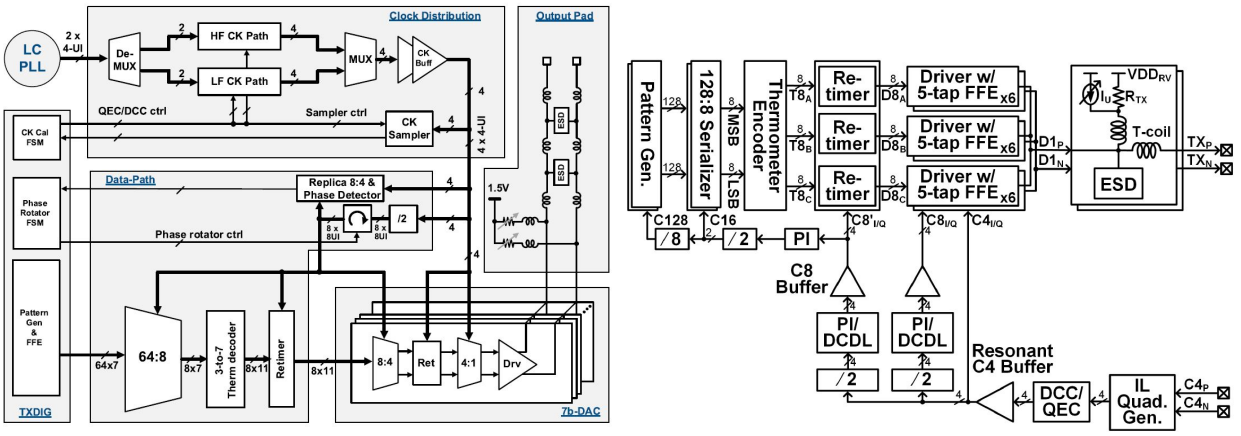
Figure 3.6: Schematic of nmos input common source (CS) amplifier

In the Intel 16 nm process, the fastest nmos is the 'ultra low threshold voltage' (ulvt) variant which has an operating frequency limit of $f_T = 300$ GHz. The gate-to-drain capacitance C_{GD} is comparable to the gate-to-source capacitance C_{GS}, hence $\gamma = \frac{C_{GD}}{C_{GS}} \approx 1$. Hence

if we consider the CS amplifier of Fig 3.6 driving an identical copy of itself (fanout of 1, i.e. FO1), then the theoretical GBW is $\frac{f_T}{1+\gamma} \approx 150\,\text{GHz}$. Traditional analytical expressions for small signal gain and bandwidth are mostly suitable for frequency ranges which are at least a decade below f_T, since parasitics start to dominate the behavior at higher frequencies. Since our target Nyquist bandwidth of 80 GHz is well beyond $\frac{f_T}{10}$ and in the vicinity of the maximum available GBW of the technology, various higher order non-idealities make it extremely challenging to get any voltage gain ≥ 1 at the ultra high speed stages of the TX.

3.3 Literature Review

Let's analyze the transmitter datapath architecture from the 2 state-of-the-art designs in Table 3.1 to get some ideas about the design of the individual components.



(a) Kim et al [5] (b) Wang et al [13]

Figure 3.7: Comparison of State-of-the-Art Transmitter Architectures

Some key take-aways of the first design [5] are:

- There is a 1 UI pulse generator based 4:1 high speed multiplexer as the pre-driver stage using quadrature rate clocking.
- The separate output driver stage employs a 9^{th} order Bessel LC filter to equalize the necessary ESD capacitors before connecting to the output pads.
- The 8 tap FFE is implemented entirely in digital PnR flow rather than AMS.

Some key take-aways of the second design [13] are:

- The 1 UI pulse generator based 4:1 high speed multiplexer using quadrature rate clocking is the output driver stage.

- A simple t-coil is used to equalize the large parasitic capacitance ($\sim$ 420 fF) at the output pad.
- The 5 tap FFE is implemented in the AMS domain.

3.4 Proposed Transmitter Datapath Architecture

With the learnings from the literature review and a few other considerations, the following decisions were adopted for the 160 Gbps transmitter design:

1. The same clocking network architecture will be shared in the transmitter and the receiver. This implies that the serialization ratio in the TX has to match the deserialization ratio in the RX.
2. For the test-chip, 20 GHz differential clocks will be taken as external input rather than having a complete on-chip PLL. This slightly simplifies the design challenges for the test chip by avoiding the extra jitter from the PLL in addition to the existing jitter from supply and clock distribution.
3. The external differential 20 GHz clocks go into an octature generator and will also be divided down to the necessary lower clock frequencies. Because of this, the high speed multiplexer in the TX has to be 8:1 instead of 4:1. A 4:1 multiplexer (mux) outputting 160 Gbps NRZ data stream would require quadrature 40 GHz clocks which would be difficult to reliably generate and distribute on-chip. While this decision simplifies the clock network, it further complicates the TX datapath design.
4. Using a 1 UI pulse generator based 8:1 high speed mux as the final driver stage would make the parasitic capacitances at the output node too large to be equalized, so instead there will be a separate 8:1 mux pre-driver stage and a final output driver stage. The design and optimization of this high speed 8:1 mux is explored in more details in the Chapter 4.
5. The ESD requirements at the output pads can be relaxed from the usual CDM rules, because this transceiver is targeting ultra short reach on-package D2D applications, and the TX output or RX input will never be exposed to off-chip connectors. This slightly relaxes the load capacitance of the output driver stage. Since designing a complete 9^{th} order Bessel LC filter is extremely difficult, we instead used co-planar waveguide (CPW) and an asymmetric t-coil to equalize the load capacitance.
6. The two FFE taps are implemented in AMS domain using a novel strategy involving the octature clock phases, as explored in more details in Chapter 5.

The complete transmitter datapath architecture is shown in Fig 3.8.
The various components of the datapath are explained below.

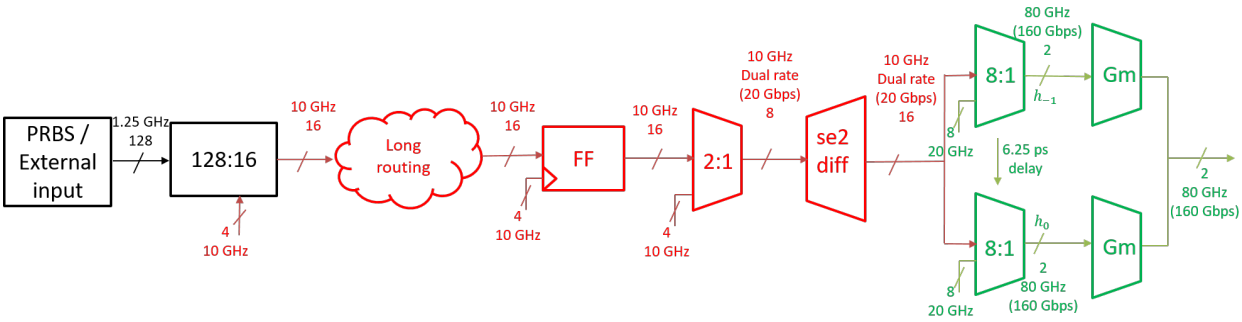


Figure 3.8: Complete Transmitter Datapath Architecture

1. The input to the datapath is a parallel bus of width 128, with each lane running at 1.25 GHz, coming from an on-chip PRBS (pseudo random bit sequence) generator or some user defined data pattern provided using the scan chain.

2. The input goes into 16 shift register based 8:1 serializers, clocked by quadrature 10 GHz clocks. The 16 output bits are retimed using an array of flip-flops after some long routing, and then go through a 2:1 mux to create 8 dual rate 20 Gbps data streams. These then go through an array of single-ended to differential converters (se2diff) to create 16 differential 20 Gbps data streams.

3. The 16 differential 20 Gbps data streams go to two high speed 8:1 muxes to create the differential 160 Gbps data streams for h_0 (main cursor) and h_{-1} (pre-cursor) by utilizing the octature 20 GHz clocks. The cursor selection scheme at this interface is elaborated in Chapter 5, and the high speed 8:1 mux design is presented in Chapter 4.

4. The two FFE cursors are combined using the final output driver stage and sent to the output pads through CPW and t-coils.

5. All the blocks colored in black deal with 1.25 GHz data. Blocks colored in red deal with 10 GHz data, while those in green deal with 20 GHz or higher frequency data.

The complete transmitter clock network architecture, designed by my collaborator Yi-Hsuan Shih, is shown in Fig 3.9, and follows the same color encoding as Fig 3.8. As mentioned before, external clock input is taken in this test chip to avoid the design effort of making an on-chip PLL, which will exist in a real system. This thesis will not cover details about the design and optimization of the clock network components.

The combined layout of the TX datapath and clock network, fully generated and measured by BAG 3++, is shown in Fig 3.10 with annotations of the datapath and clock components from figures 3.8 and 3.9. The top level power grid is not shown in the figure for better legibility.

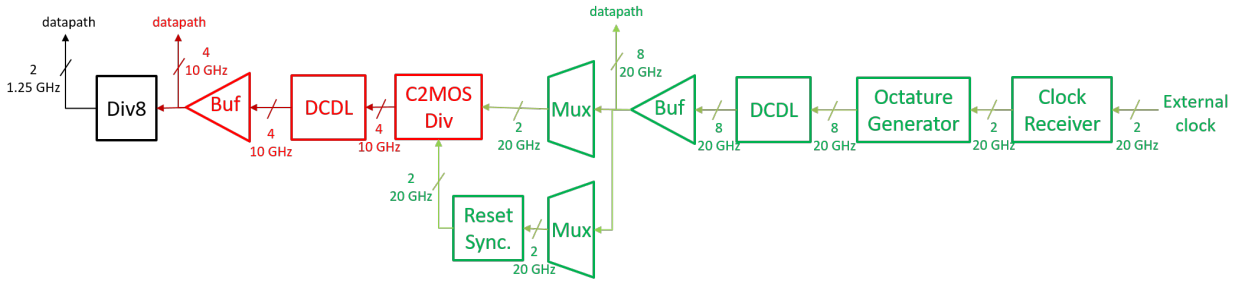


Figure 3.9: Complete Transmitter Clock Network Architecture

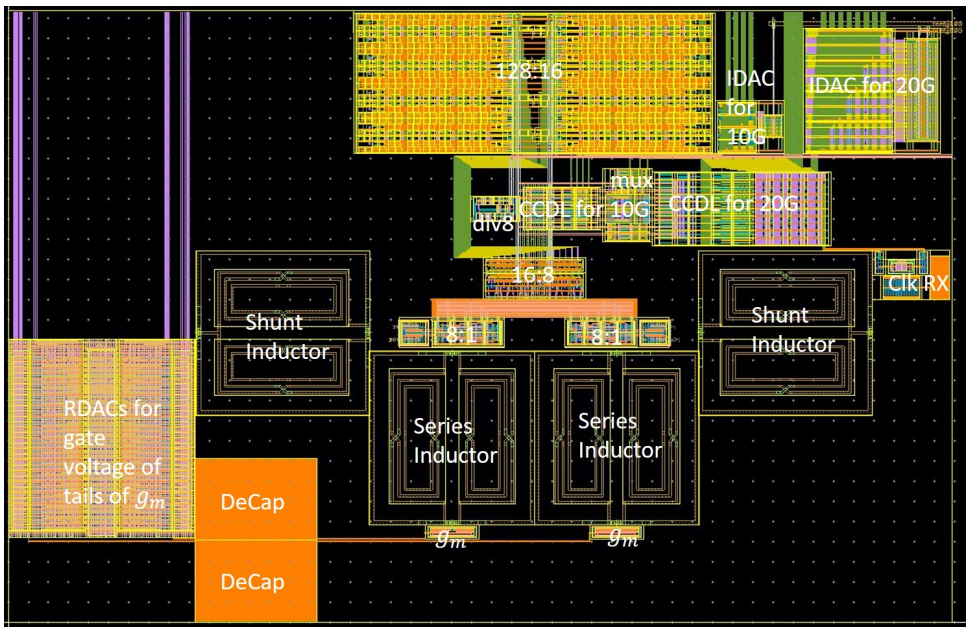


Figure 3.10: BAG 3++ Generated Transmitter Layout with annotated components

The complete TX datapath top level, with CPW and t-coils designed by my collaborator Kunmo Kim, is shown in Fig 3.11. Top level power grid is again removed for better visibility. This thesis will not cover details about the design and optimization of the CPW and t-coils.

The die photograph is shown in Fig 3.12. The transmitter datapath is marked in a green box while the receiver (RX) datapath is marked in a red box, showing the GSSGSSG bumps needed for RX outputs and TX inputs at the edge of the die. The extra bumps to the left of the red box and to the right of the green box are for 20 GHz external clock inputs for the RX and TX respectively.

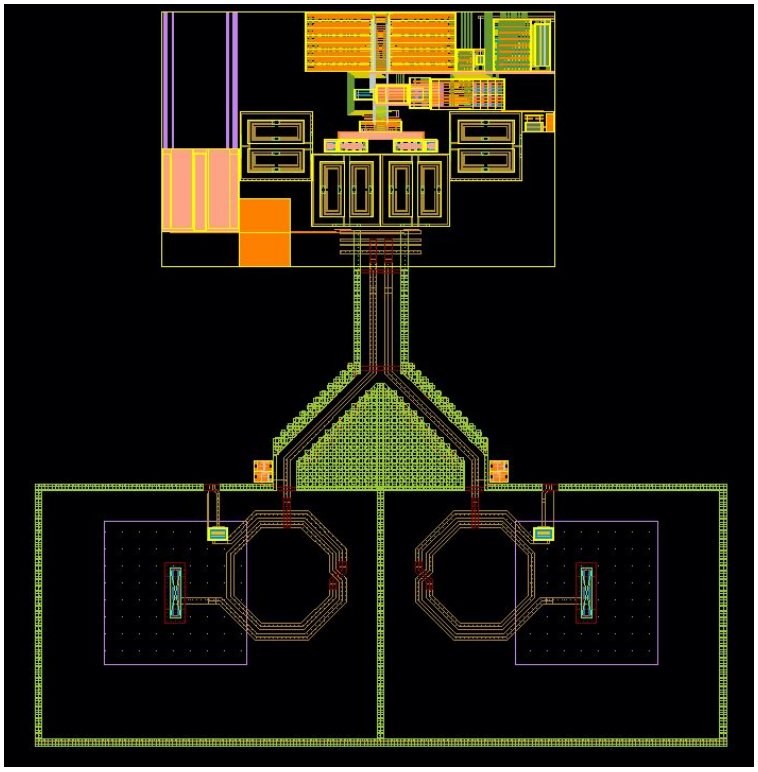


Figure 3.11: Manually Integrated Transmitter Datapath Top Level Layout

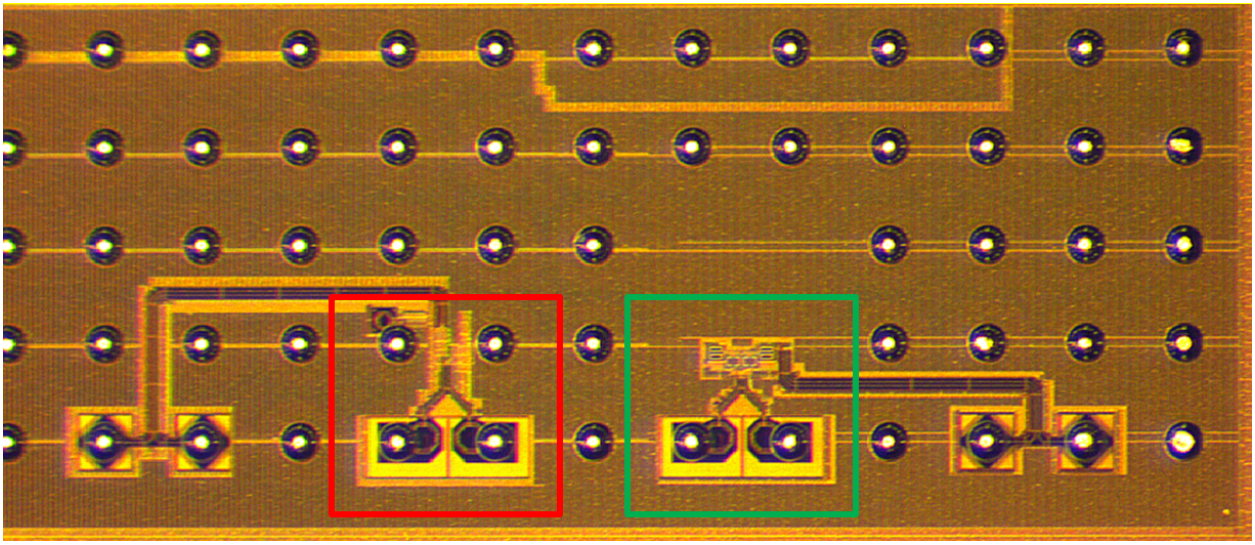
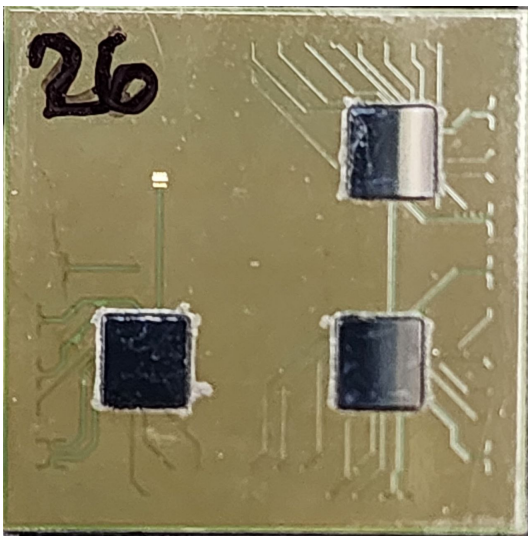


Figure 3.12: Die photograph

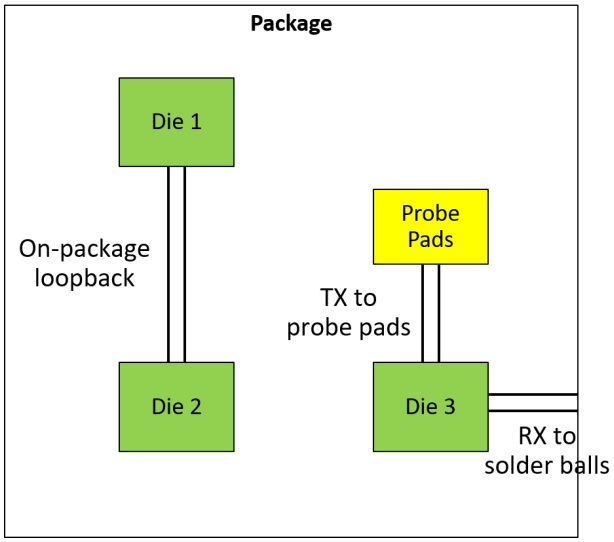
3.5 Package Overview for Testing

The organic substrate package, designed by my collaborator Bob Zhou, houses three dies to allow different testing modes of the transmitter and receiver. The balls-up overview and the real photograph of the die attached package is shown in Fig 3.13. The different testing modes are as follows:

1. **Loopback mode:** The TX on die 2 transmits data to the RX on die 1 over the 8.5 mm differential on-package channel, estimated to have ~ 3 dB loss at the Nyquist frequency of 80 GHz. The TX on die 1 and the RX on die 2 are deactivated and not tested.
2. **Stand-alone TX testing mode:** The TX outputs from die 3 can be directly probed from the probe pads, and eye diagrams can be observed in a high speed oscilloscope or signal analyzer.
3. **Stand-alone RX testing mode:** External signal generators can be used to feed data input directly to the RX on die 3 through the solder balls using W (1.85 mm) connectors on the PCB.



(a) Photograph of Die attached Package



(b) Balls-up overview of the MCP

Figure 3.13: Organic Substrate Package

The PCB design and lab measurements are elaborated in Chapter 7.

Chapter 4

Design Optimization of High Speed 8:1 Multiplexer

4.1 Motivation

As discussed in Chapter 3, the motivation for the high speed 8:1 multiplexer (mux) can be summarized as follows:

- A 4:1 mux for 160 Gbps data-rate NRZ operation would require quadrature 40 GHz clock phases, which is extremely difficult to reliably generate and distribute across PVT variations, device mismatches, etc.
- Instead, the clock network provides octature 20 GHz clock phases, so the high speed mux stage has to be 8:1.

The inputs for the 8:1 mux are 16 differential 20 Gbps data streams which are clocked using the octature 20 GHz phases. There are two major sub-parts for the mux design:

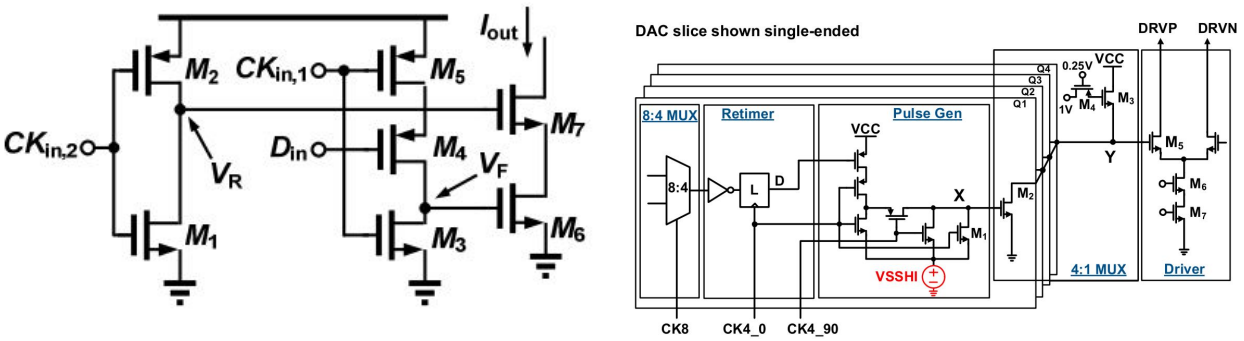
1. Designing the 1 UI pulse generator
2. Designing the peaking architecture for extending the bandwidth

4.2 1 UI Pulse Generator Design and Optimization

Let's analyze the 1 UI pulse generator architectures used in literature for high speed 4:1 mux designs. Fig 4.1 shows two different architecture styles.

Some key takeaways of the first design [2] are:

- The current pulse I_{out} occurs, i.e. the data input D_{in} is sampled, when both input clock phases $CK_{\text{in,1}}$ and $CK_{\text{in,2}}$ are low.



(a) Chen et al [2] (b) Kim et al [5]

Figure 4.1: 1 UI Pulse Generator Architecture Options

- The rising edge of V_R, i.e. the falling edge of $CK_{\text{in},2}$, independently controls the rising edge of the current pulse I_{out}.
- The falling edge of V_F, i.e. the rising edge of $CK_{\text{in},1}$, independently controls the falling edge of the current pulse I_{out}.
- If transistors M6 and M7 are in saturation, then the output resistance is given by $R_{\text{out}} \approx g_{m7} r_{o7} r_{o6}$.

Some key takeaways of the second design [5] are:

- The current pulse I_{out} occurs, i.e. the data input D is sampled, when both input clock phases $CK4\_0$ and $CK4\_90$ are low.
- The falling edge of $CK4\_90$ independently controls the rising edge of the current pulse I_{out}.
- The rising edge of $CK4\_0$ independently controls the falling edge of the current pulse I_{out}.
- If transistor M2 is in saturation, then the output resistance is given by $R_{\text{out}} \approx r_{o2}$.

It is clear that the 2 architectures are conceptually similar. They use different phases of the quadrature clock to control the rising and falling edges of the 1 UI current pulse. This muxing strategy has an additional jitter attenuation advantage, as analyzed in [5]. Since the jitter on the rising edge of the current pulse is uncorrelated from the jitter on the falling edge of the pulse which occurs due to a different quadrature phase, the averaging of the uncorrelated jitter improves the differential pulse jitter by a factor of $\frac{1}{\sqrt{2}}$.

For the 160 Gbps NRZ transmitter design, the first 1 UI pulse generator architecture is chosen because of the higher output resistance, to improve the small signal gain of this high

speed mux stage. Instead of two different phases of quadrature clock in the previous designs, two different phases of octature clock will be chosen for each 1 UI pulse generator such that their 'low' states overlap for exactly 1 UI.

4.3 Peaking Topology for Bandwidth Extension

The first order RC bandwidth of eight 1 UI pulse generator units driving the output driver stage is lower than the desired Nyquist frequency of 80 GHz, hence inductive peaking topologies have to be explored to extend the bandwidth. Kim et al [5] uses active inductors to extend the bandwidth at the output node of the 4:1 mux, as shown in Fig 4.1b.

There are 2 major disadvantages for using active inductors in the 160 Gbps NRZ TX design in the Intel16 technology node.

1. In this process, the active inductor has a self resonant frequency lower than 80 GHz, and essentially becomes a capacitor at the higher frequencies.

2. Active inductors inherently increase bandwidth by trading off gain, since the impedance looking into the source of M3 in Fig 4.1b is given by $Z_{\text{in}} \approx \frac{1}{g_{m3}} + \frac{sR_{\text{on},4}}{\omega_{T3}}$. As explained in the theoretical analysis in Chapter 3, it is already extremely challenging to get a gain ≥ 1 at the high speed stages because of the technology parameters. Hence trading off the gain even further using active inductors is not a viable option.

This implies that passive on-chip inductors have to be used for the peaking topology. The new challenge is dealing with the large parasitic capacitance of these inductors which can significantly reduce the perceived bandwidth extension advantage by lowering the self resonant frequency.

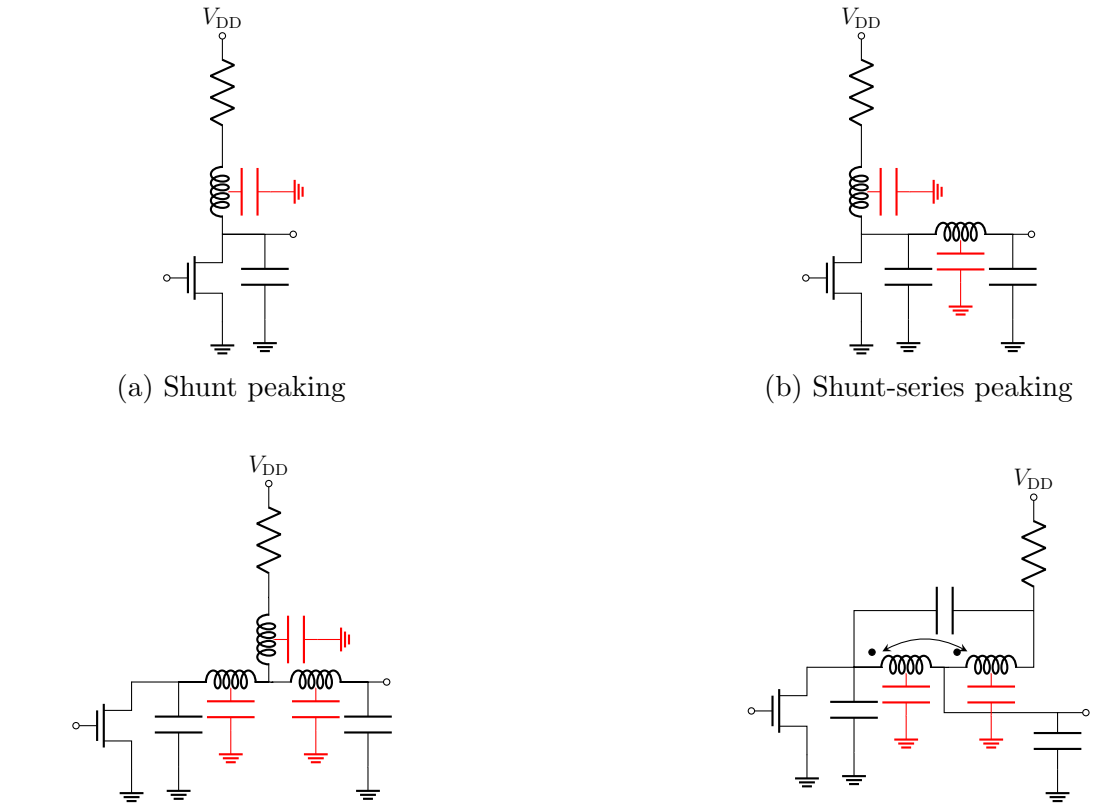
Literature Review of Passive Inductive Peaking Topologies

The four most common inductive peaking topologies are represented in Fig 4.2 taken from [4] with some capacitor annotations.

Extensive theoretical analysis about these topologies from [7], [8], [11], [9], [10] can be summarized as follows:

- Without peaking, $BW = \frac{1}{RC}$
- With inductive shunt peaking, $BW = \frac{1.8}{RC}$
- With symmetric t-coil peaking, assuming damping factor of $\frac{1}{\sqrt{2}}$, $BW = \frac{2.83}{RC}$

At first glance, it may seem like t-coil peaking is the best peaking option for all applications, but that is misleading, and we need to fully understand the details. One major drawback of the theoretical analysis is that it often ignores the parasitic capacitances associated with the



(a) Shunt peaking

(b) Shunt-series peaking

(c) Shunt and double series peaking

(d) Bridged t-coil peaking

Figure 4.2: Common Peaking Topologies

metal wires of the inductor or t-coil, which are annotated in red in Fig 4.2. As the inductor shapes and sizes, coupling ratio between multiple inductors, etc. is modified based on theoretical analysis, the parasitic capacitances change as well and can alter the performance resulting in very different results from what was predicted by theory. In older technology nodes, device parasitics used to be 1 or more orders of magnitude larger than wire parasitics, so this omission resulted in negligible errors. However, in advanced nm scale FinFET technology nodes, device scaling has made their parasitics comparable to wire parasitics, so it becomes extremely crucial to take the wire parasitics effect into account. In some cases, adding the inductors may even do more harm than good if the parasitic capacitances completely nullify any bandwidth extension advantages and instead end up increasing the load capacitance more than the initial value. One common example application where a practical design is more important than a theoretical design, is the use of asymmetric t-coils instead of symmetric t-coils for equalizing the ESD at the pad connections. These asymmetric t-coils are designed through multiple simulation iterations and trial-and-error since theoretically analyzing the behavior becomes impractically difficult.

The small signal bandwidth may not always be the main objective function that needs to be maximized for every application. Since the bandwidth extension using peaking topologies always comes at the expense of increased latency, some other factors may become more crucial objective functions instead.

The effectiveness of the different peaking topologies also depends on the distribution of the load capacitances across the inductor network. [4] shows that a bridged t-coil is most effective when the output node has the largest lumped capacitance, while the same t-coil with opposite polarity coupling coefficient is more preferable if the driver side capacitance starts to dominate.

[4] also shows examples where the objective function for a particular application may be something other than just the small signal bandwidth. A CML (current mode logic) buffer or amplifier stage, that works predominantly in the small signal mode of operation, should be optimized for maximally flat group delay response, which is usually achieved using the bridged t-coil topology. On the other hand, a logic stage like a CML latch should be optimized for minimum settling error since it exhibits large signal mode operation with rail-to-rail input and output swings, which can be achieved using the shunt and double series peaking topology.

Another important factor while choosing a peaking topology is the optimization of the entire system as a whole rather than optimizing every single sub cell. Again, as shown in [4], a 2:1 CML multiplexer with shunt peaking on the first stage and shunt and double series peaking on the second stage provided the best overall performance of the multiplexer rather than optimizing each stage in isolation.

All these considerations convince us that the most reliable way of designing inductive peaking topologies is by running some automated program that can leverage computer resources to perform iterations and optimize the application specific objective function. While [4] presents their design optimization routine for on-chip inductive structures, we have presented our open source, automated, process portable electromagnetic and circuit co-optimization routine with BAG 3++ to handle this exact problem in Chapter 2.

4.4 Optimizing the Design of the 8:1 mux

After running many experiments, the following lessons were learned, and helped to prune the search space for the co-optimization routine to arrive at the optimal design:

1. For any inductive peaking topology that has a shunt inductor branch, e.g. cases a, b, c in Fig 4.2, it is better to swap the position of the inductor and the resistor. Then the resistor isolates the parasitic capacitance of the inductor (which is usually higher than the parasitic capacitance of the resistor) from the high speed signal nodes.

2. It can be better to avoid using the top metal layer for the inductor or t-coil, and instead use the second layer from the top, for a few reasons:

- The second layer may have a smaller minimum allowed width from the design rules, which results in lower substrate capacitance as compared to the top layer which usually has larger minimum allowed width, despite the larger distance to the substrate. This effect was observed in [4] as well.
- The second layer may be thinner than the much thicker top metal layer, resulting in lower side wall capacitance between inductor turns on the second layer.

These factors depend heavily on the exact metal stack in the technology node, and need to be verified from simulations and measurements based on the PDK characteristics.

3. The objective function for the 8:1 mux design is maximizing the eye height at the input of the output driver stage. It was found that the shunt-series topology provided the best performance for this application. The schematic of the 8:1 mux stage with shunt-series peaking and driving the final Gm stage, is shown in Fig 4.3. The coloring follows the style from Fig 2.1 and Fig 2.2, where the circuits in the blue boxes form the sub DUT for parasitic extraction, whereas the circuits in the yellow boxes form the sub DUT for electromagnetic simulation.

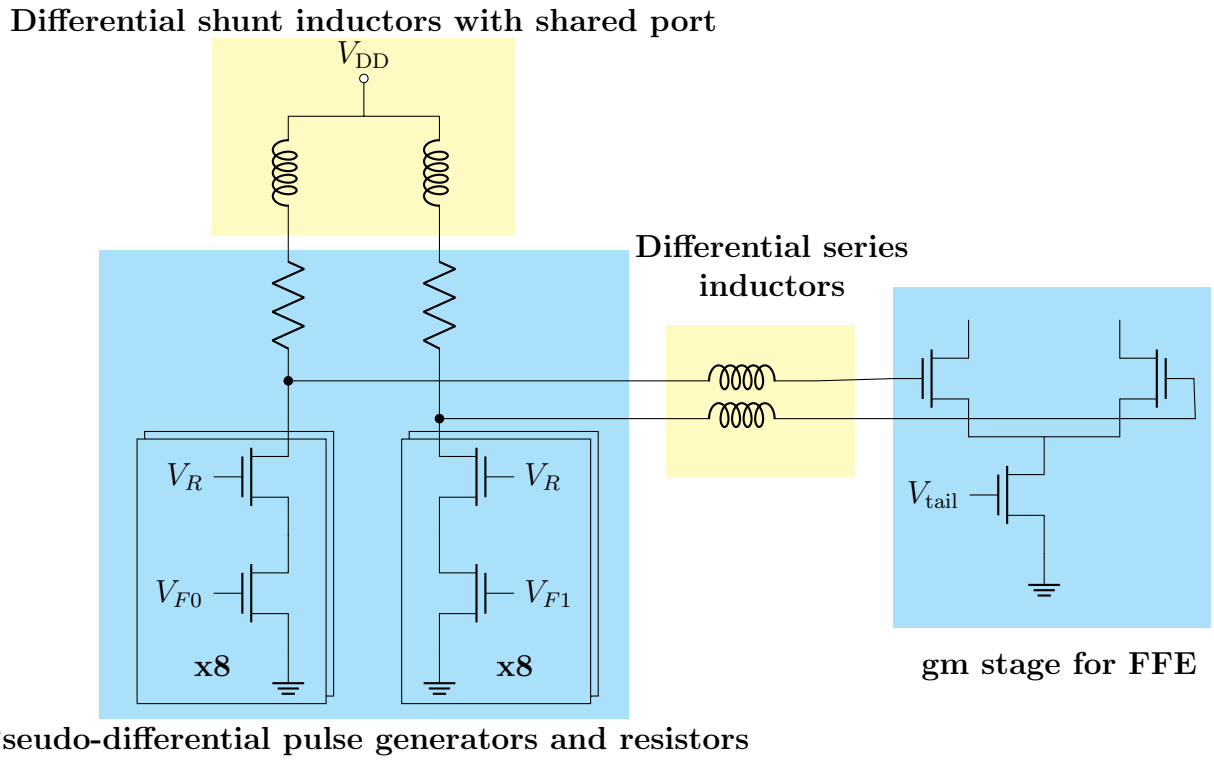
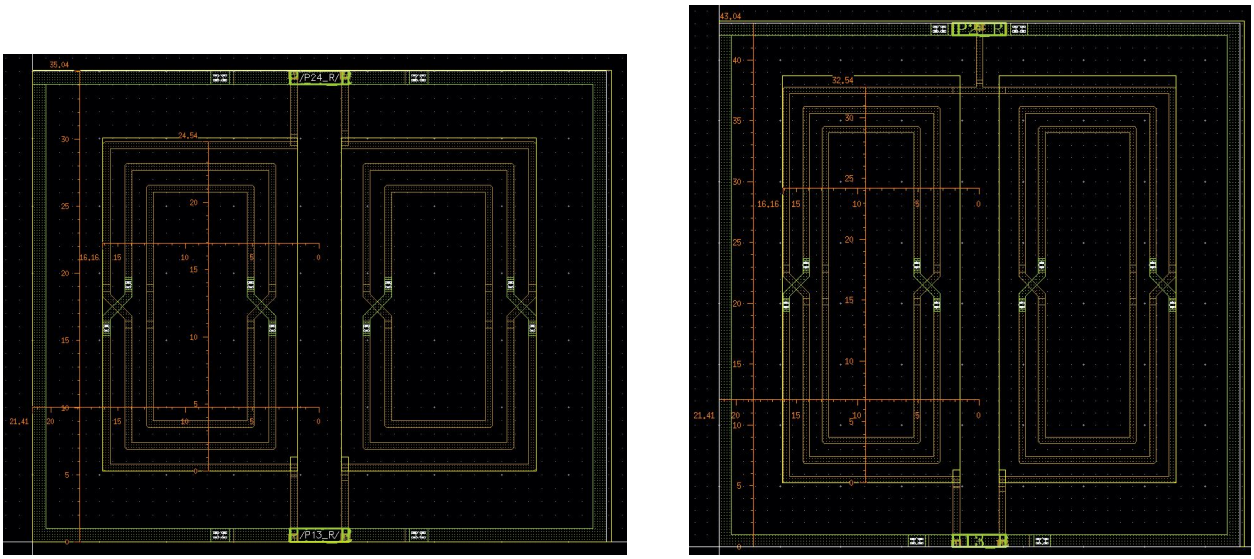


Figure 4.3: Schematic of 8:1 multiplexer with shunt-series peaking inductors

4. To optimize the area of the overall transmitter floorplan in Fig 3.10, rectangular inductors are used instead of octagonal to achieve maximum inductance in minimum area. The inductors are tapped at the top and bottom edges to reduce the length of the routing wires to the leads, thereby minimizing the additional parasitic capacitances from routing. Fig 4.4 shows the optimized layouts of the differential series and shunt inductors on the GM0 layer.



(a) Differential Series Inductors

(b) Differential Shunt Inductors

Figure 4.4: Optimized BAG 3++ generated layouts of inductors for peaking of 8:1 mux

Results

The maximum differential peak-to-peak swing at the input of the output driver stage is $\sim$ 320 mV at two PVT variations (TT@25 °C with $V_{DD} = 1$ V, SS@-40 °C with $V_{DD} = 0.9$ V). At FF@125 °C with $V_{DD} = 1.05$ V, the swing reduces to $\sim$ 240 mV, possibly because of lower output resistance of the 1 UI pulse generators. We expect $> 99.9\%$ yield at TT@25 °C from the Intel foundry.

Chapter 5

Datapath Timing at Clock Crossing Interfaces

5.1 Problem setup

The most challenging timing in the transmitter datapath happens at the input of the high speed 8:1 muxes. There are 8 data streams at dual rate 10 GHz (i.e. 20 Gbps NRZ) at the end of the complete 128:8 serialization (consisting of the eight 8:1 shift register based serialization followed by 2:1 muxing in Fig 3.8) which get converted to 16 differential data streams by the single-ended to differential (se2diff) converter. These 8 data streams (and their complementary counterparts) are on different phases of the quadrature 10 GHz clocks, and become inputs to the two 8:1 muxes. The octature 20 GHz clocks are the 'select' signals for the 8:1 muxes, and have to be intelligently chosen to sample the data onto the 1 UI pulse generator circuits of the 8:1 muxes, simultaneously creating two FFE cursor streams h_0 (main cursor) and h_{-1} (pre cursor) on the two paths, where h_0 is delayed from h_{-1} by 1 UI = 6.25 ps. Hence there is a very small margin of error for aligning the octature 20 GHz clock edges with the input data streams while maintaining adequate setup and hold time margins. Not only is the logic timing quite tricky for the two 8:1 muxes, but the analog propagation delay across logic gates and routing parasitics also need to be taken into account for the data and clock edge alignment.

5.2 Alignment of Data and Clock Edges

Ideal Scenario (no propagation delay)

Let's first consider the ideal situation where there is no propagation delay (Verilog-style behavior) and figure out the logic timing to create the data streams for the 2 FFE cursors. We first need to closely analyze which of the eight 10 GHz input data streams are on which phases of the quadrature 10 GHz clocks.

Since this clock crossing interface is the most challenging part of the transmitter datapath with regards to timing margins, we start by assigning the quadrature clock phases to the data streams in the way that works best for this interface. Then we configure the clock phases on all the previous stages of the serializer to enforce the desired behavior.

Let's name the eight 20 Gbps data streams as d_0 to d_7. The complementary eight data streams will follow the same logic, so we focus on only the true data phases for now. The quadrature 10 GHz clock phases will be referred to as clk10_0, clk10_{90}, clk10_{180}, and clk10_{270}. Since the last two phases are complementary versions of the first two, we will focus on only clk10_0 and clk10_{90} in the subsequent discussion.

The octature 20 GHz clock phases will be referred to as clk20_0, clk20_{45}, ..., clk20_{315}. Phase rotators and reset synchronizers in the clock network in Fig 3.9 ensure that the rising edge of clk20_0 aligns with the rising and falling edges of clk10_0 and clk10_{180}.

Assign d_0, d_1, d_2, d_3 to the clock phase clk10_0, i.e. these four dual rate data streams update on both rising and falling edge of clk10_0. Similarly, d_4, d_5, d_6, d_7 are assigned to the clock phase clk10_{90}, i.e. these four dual rate data streams update on both rising and falling edge of clk10_{90}. This is illustrated in Fig 5.1, where the red clock waveforms are at 10 GHz and the green clock waveforms are at 20 GHz, following the color convention from Fig 3.8.

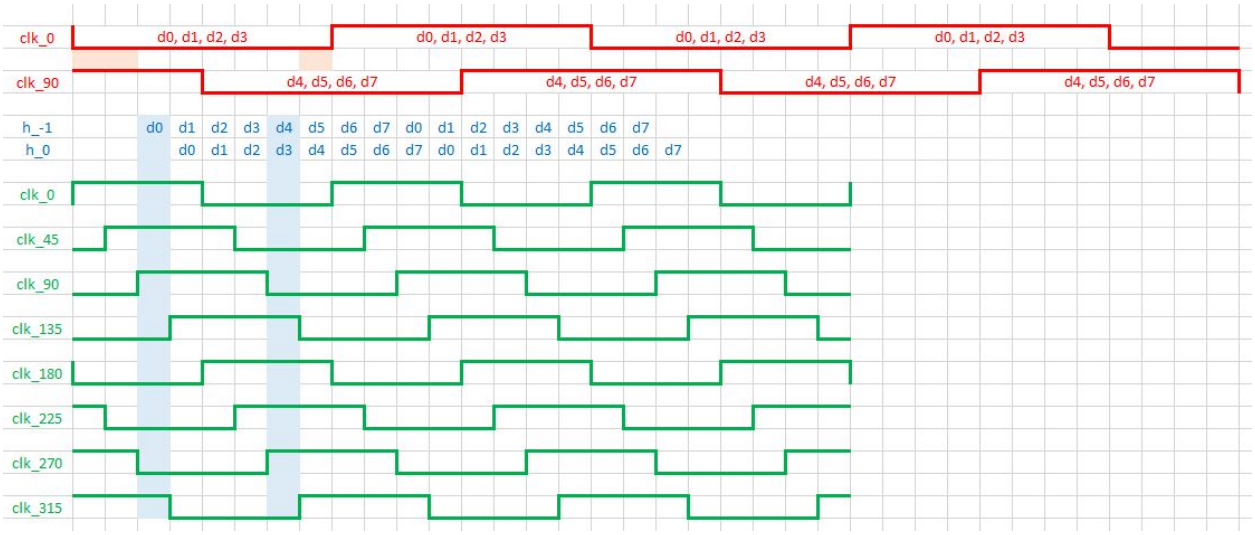


Figure 5.1: Timing diagram at clock crossing interface: Ideal scenario

Now let's consider the ideal alignment of the h_{-1} and h_0 cursors relative to the quadrature 10 GHz clock phases. Assign d_0 to h_{-1} 2 UI after the first falling edge of clk10_0. In Fig 5.1, this is referring to the UI corresponding to the first blue shaded column. This means that there is a 2 UI setup time margin for d_0 to be sampled by the 8:1 mux for h_{-1} after it became available at a clock edge. The 8:1 mux for h_0 then samples d_0 in the next UI when h_{-1} is sampling d_1.

Following this logic, d_3 is assigned to h_0 just 1 UI before it will get a new value at the first rising edge of clk10_0. In Fig 5.1, this is referring to the UI corresponding to the second blue shaded column. This means that there is a 1 UI hold time margin for d_3 to be sampled by the 8:1 mux for h_0 before it gets refreshed by the next clock edge.

By the exact same argument, d_4 is sampled by the 8:1 mux for h_{-1} 2 UI after it became available at the first falling edge of clk10_{90}, and d_7 is sampled by the 8:1 mux for h_0 1 UI before it gets refreshed by the first rising edge of clk10_{90} in Fig 5.1. Hence this scheme provides 2 UI setup time and 1 UI hold time margin for creating h_0 and h_{-1} from the eight input data streams.

In Chapter 4, it was mentioned that each 1 UI pulse generator in the 8:1 mux samples the input data when both of its clock inputs are 'low'. Based on that information and the green octature waveforms in the timing diagram, we can infer that d_0 is sampled for h_{-1} by the pair of clk20_{135} and clk20_{270} in the UI represented by the first blue shaded column. In the UI represented by the second blue shaded column, the pair of clk20_{315} and clk20_{90} samples d_4 for h_{-1} and d_3 for h_0.

Working through all the other UIs, we can construct Table 5.1 to show which 20 GHz clock phase pairs sample all the data streams $d_0, \ \ldots, \ d_7$ for the main cursor and pre cursor.

Table 5.1: Pairs of 20 GHz clock phases to create FFE cursor data streams (ideal)

	h_{-1}	h_0
d_0	clk20_{135}, clk20_{270}	clk20_{180}, clk20_{315}
d_1	clk20_{180}, clk20_{315}	clk20_{225}, clk20_{0}
d_2	clk20_{225}, clk20_{0}	clk20_{270}, clk20_{45}
d_3	clk20_{270}, clk20_{45}	clk20_{315}, clk20_{90}
d_4	clk20_{315}, clk20_{90}	clk20_{0}, clk20_{135}
d_5	clk20_{0}, clk20_{135}	clk20_{45}, clk20_{180}
d_6	clk20_{45}, clk20_{180}	clk20_{90}, clk20_{225}
d_7	clk20_{90}, clk20_{225}	clk20_{135}, clk20_{270}

Hence we have determined the correct clock and data alignment for the ideal scenario with no propagation delay. It is worth noting that this scheme cannot be indefinitely extended for more number of FFE taps, since we will run out of UIs for setup and hold time margins.

Real scenario

Based on simulations of extracted netlists, it is found that there is an average of ∼24 ps clk-to-q delay from a rising or falling edge of the quadrature 10 GHz clock phase at the 2:1 mux stage to the data settling to a steady value at the output of the se2diff stage, with a

$\sim$1.8 ps spread across PVT variations. Since 1 UI = 6.25 ps, 24 ps $\pm$ 0.9 ps corresponds to a 4 UI propagation delay until the data $d_0, \ \ldots, \ d_7$ is ready to be sampled after a rising or falling edge of clk10_0 or clk10_{90} in the timing diagram of Fig 5.1.

This implies that the pair of 20 GHz clock phases that selected d_4 originally, will now select d_0 instead. Based on this, we can reconstruct Table 5.2 to show which 20 GHz clock phase pairs sample all the data streams $d_0, \ \ldots, \ d_7$ for the main cursor and pre cursor, once propagation delay is taken into account, by shifting the contents of Table 5.1 by 4 UI.

Table 5.2: Pairs of 20 GHz clock phases to create FFE cursor data streams (real)

	h_{-1}	h_0
d_0	clk20_{315}, clk20_{90}	clk20_0, clk20_{135}
d_1	clk20_0, clk20_{135}	clk20_{45}, clk20_{180}
d_2	clk20_{45}, clk20_{180}	clk20_{90}, clk20_{225}
d_3	clk20_{90}, clk20_{225}	clk20_{135}, clk20_{270}
d_4	clk20_{135}, clk20_{270}	clk20_{180}, clk20_{315}
d_5	clk20_{180}, clk20_{315}	clk20_{225}, clk20_0
d_6	clk20_{225}, clk20_0	clk20_{270}, clk20_{45}
d_7	clk20_{270}, clk20_{45}	clk20_{315}, clk20_{90}

Any additional UI shifts, due to PVT variations or clock phase misalignment, can be calibrated using the 4 UI phase rotators, denoted by the 2 muxes before the C2MOS divider in the TX clock network architecture in Fig 3.9.

5.3 Verification of clock and data alignment from extracted simulations

It is extremely crucial to verify the octature 20 GHz clock phase selection using rigorous circuit simulations. While behavioral modelling can help to verify the logic in the ideal scenario, only extracted simulations can verify the behavior taking the analog propagation delay into account.

The most rigorous and reliable verification method is to run a transient simulation of the entire extracted transmitter with datapath and clock network. However, that will make the simulation time be prohibitively long, so a more efficient alternative approach is necessary that has a more reasonable simulation time without sacrificing the rigor and reliability of the verification step.

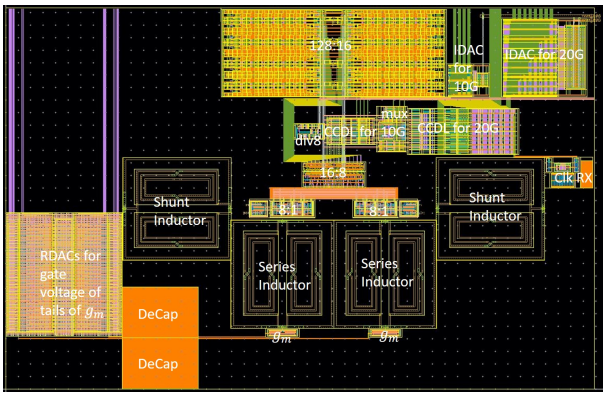
One approach is to extract the sub cells in isolation, and then tying them up in the testbench (DUT and harnesses feature in BAG 3++ described in Chapter 1), with additional

lumped resistors and capacitors to model the parasitics because of routing between the sub cells. The Intel16 PDK provides pcells (e.g. `rcint` in the `intel22prim` library) that model the distribution of parasitic capacitance and resistance of various metal layers with user specified widths, lengths, surrounding metal grid, etc. While this is a great strategy during the initial design stage when the top level layout has not been assembled yet, it is still attempting to predict the routing parasitics of the real design.

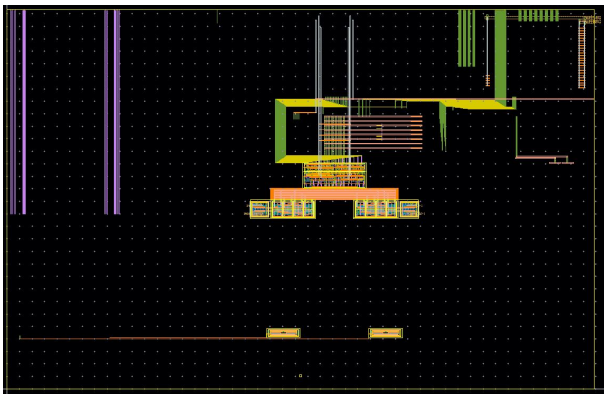
The partial layout generation feature of BAG 3++ provides the most reliable way of running these verification simulations. In Chapters 2 and 4, this feature was used to divide a DUT layout into a magnetic passives only sub DUT for electromagnetic simulations and another non-magnetic sub DUT for parasitic extraction. Now we see the second major application of this feature. The layout of the entire transmitter can be generated by omitting all the components except the ones whose interface is being analyzed, while still retaining all the top level routing and power grid. This ensures that the extracted netlist captures the true parasitics of the routing at the interface of the sub cells, and yet have reasonable simulation times by excluding other subcells that are not crucial for the particular interface verification.

Example 1: Datapath Timing Verification at Clock Crossing Interface

Fig 5.2 shows the partial layout generated for this datapath timing verification step. The only subcells that are generated are the 2:1 mux stage, se2diff stage, and high speed 8:1 mux stages from the transmitter datapath architecture in Fig 3.8, and all the top level interface routing is preserved. The power grid is not shown in the figure for visibility, but is present for the actual simulation.



(a) Complete Transmitter Layout from Fig 3.10



(b) Partial layout

Figure 5.2: Partial Layout for Datapath Timing Verification at Clock Crossing Interface

The testbench and the measurement details of this timing verification application are summarized as follows:

1. **Inputs:**
    - **Clock:** Instead of square wave clock waveforms, buffered sinusoidal waveforms are used for both the quadrature 10 GHz and octature 20 GHz clocks to account for worst case low pass filtering from the clock distribution network.
    - **Data:** 16 constant data input streams are provided as input to the 8 bit 2:1 mux. This is assuming that the 16 outputs of the earlier low speed 128:16 serializer have been retimed by the flop stage after the long routing, and have settled before they get sampled onto the 2:1 mux.
2. **Outputs:** output data streams of the two high speed 8:1 muxes
3. **Verify:** Across all PVT variations, the following should hold true:
    - The output data pattern should correctly match the muxed / serialized version of the 16 input data streams for both the high speed 8:1 muxes
    - The output data stream from the 8:1 mux for h_0 should be exactly 1 UI delayed from the corresponding data stream for h_{-1}

This measurement reliably verifies the operation of the novel timing technique for creating the two data streams for the FFE taps, as introduced in Fig 5.1 and Table 5.2.

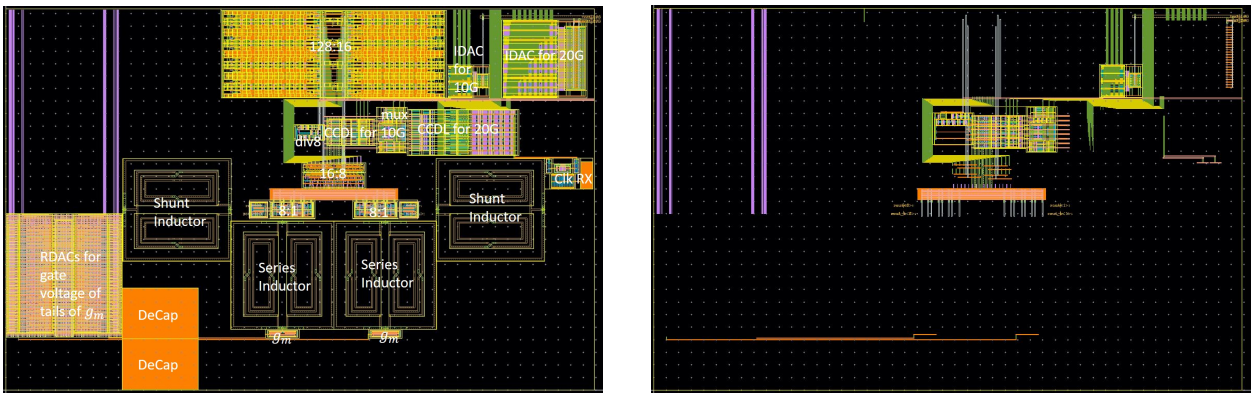
Example 2: Verification of Clock Phase Alignment

It is important to verify that the quadrature 10 GHz and octature 20 GHz clock phases can be aligned correctly across PVT variations using the available calibration options via the phase rotator. As mentioned earlier, the rising edge of clk20_0 must align with the rising and falling edges of clk10_0 and clk10_{180} for the timing technique in Fig 5.1 to operate successfully.

Fig 5.3 shows the partial layout generated for this verification. The only generated subcells are the C2MOS divider, corresponding CCDL (current controlled delay line) and buffers, and the div8 stage from Fig 3.9, and all the top level interface routing is preserved. The power grid, not shown in the figure for visibility, is present for the actual simulation.

The testbench and measurement details for this application are summarized as follows:

1. **Inputs:**
    - **Clock:** Instead of square wave clock waveforms, buffered sinusoidal waveforms are provided for the octature 20 GHz clocks to account for worst case low pass filtering from the clock distribution network.
    - **Control:** Reset signal and current DAC control signals are provided for the CCDL and div8 stages.



(a) Complete Transmitter Layout from Fig 3.10 (b) Partial layout

Figure 5.3: Partial Layout for Datapath Timing Verification at Clock Crossing Interface

2. **Outputs:**
   - Quadrature 10 GHz clocks
   - 1.25 GHz clock
3. **Verify:** Across all PVT variations, it should be possible to achieve the desired clock phase alignment using the available calibration knobs.

This measurement reliably verifies the alignment of the clock phases at the interface of the clock network and the datapath of the transmitter.

Chapter 6

Design and Verification of Miscellaneous TX Datapath Components

Chapters 4 and 5 discussed the design and optimization of two of the most challenging components of the 160 Gbps NRZ transmitter datapath. This chapter covers the design challenges and verification procedures of some of the other equally important components in the datapath.

6.1 Verification of Output Driver Stage Design

Problem setup

The output driver stage has to be designed and optimized simultaneously with the coplanar waveguide (CPW) and t-coils at the output pad of the transmitter (TX), along with the passive front-end at the input of the receiver (RX). The signal transmitted by the driver needs to travel across the on-package channel and preserve a large enough amplitude at the input of the track-and-hold (T&H) circuits at the receiver. Hence the target specifications of the driver are set by the channel characteristics and the RX T&H design, as well as link model analysis, making this design optimization step a collaborative effort with the passive frontend and the receiver designers.

Chapter 3 mentions that the transmitter will be tested in both the loopback mode as well as the stand-alone TX mode by direct probing of the output. However, the output driver stage is optimized specifically for the loopback mode test, since that is the main target application of the entire 160 Gbps NRZ transceiver project.

Testbench Setup

Fig 6.1 shows the testbench for measuring the performance of the G_m cells of the output driver. The TX components are enclosed in a green box and the RX components are enclosed in a red box, following the color coding scheme from Fig 3.12.

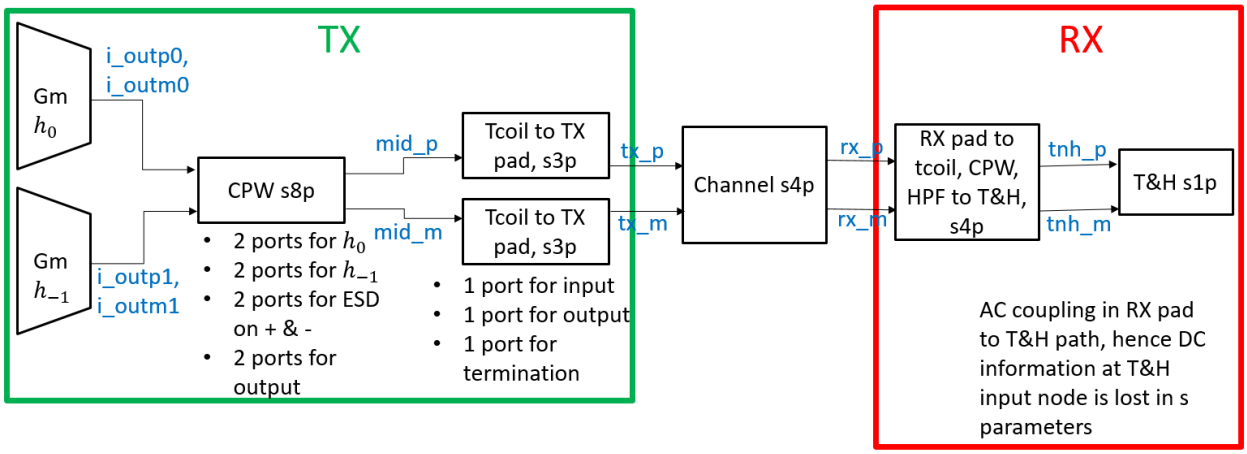


Figure 6.1: Testbench for Transmitter Output Driver Design

The different components of the testbench are explained below, starting from the right:

1. The final load is a 1 port s parameter file that models the input impedance of the RX T&H circuits, designed by my collaborator Paul Kwon. The objective function of the output driver design is to maximize the differential amplitude at the T&H inputs `tnh_p` and `tnh_m`.

2. The block before the T&H is a 4 port s parameter file that models the entire passive front-end of the receiver, including the routing of the input pads through the t-coil, CPW, high-pass filter (HPF) to the T&H inputs. Note that the presence of a HPF implies that the RX is AC coupled, so the DC bias point at `tnh_p` and `tnh_m` are set by local bias connections. Since s parameters are inherently small signal modelling at a specific bias point, the information about the exact DC biasing point at the T&H inputs is not captured in this testbench. This is ok for the purpose of this simulation since we are only interested in the differential amplitude at `tnh_p` and `tnh_m`. The inputs to this block are `rx_p` and `rx_m`.

3. The channel is modeled by a 4 port s parameter file derived by HFSS electromagnetic simulation of the 8.5 mm differential channel on the package, with ~3 dB loss at the Nyquist frequency of 80 GHz, as shown in Fig 3.13. The inputs to the channel are `tx_p` and `tx_m` which are transmitted by the TX.

4. Instead of lumping the entire passive network on the TX side into a single 4 port s parameter file, similar to what's done on the RX, we extract separate 8 port and 3 port s parameter files from electromagnetic (EM) simulations of the CPW and t-coil respectively. `mid_p` and `mid_m` are the connections from the CPW to the 2 t-coils. Extensive simulations confirm that the CPW and the differential t-coils are designed to have proper isolation, so the individual EM simulations are not losing any accuracy by not modelling the mutual coupling. By creating individual s parameter files for the passive network sub components, we can place the extracted netlists of the termination resistors and ESD in the testbench (not shown in Fig 6.1), and optimize their values and location as well. All the magnetic passives (CPW and t-coil) on the TX and RX passive front-end were designed by my collaborator Kunmo Kim.

5. The parasitic extracted netlist of the G_m cells for the 2 FFE taps is the main DUT in this testbench. The current outputs `i_outp0` and `i_outm0` for h_0, and current outputs `i_outp1` and `i_outm1` for h_{-1} are summed in the CPW with appropriate sign. Traditionally the G_m cell for the precursor tap h_{-1} is designed to be smaller than the G_m cell for the main cursor tap h_0. However, in this design, the same sized G_m cell is used for both to save some design time and effort. The FFE co-efficient magnitude is programmed by adjusting the gate voltage of the tail transistor in G_m cell using voltage DACs. This also allows for some redundancy in the sense that one can easily program which G_m cell is used for which cursor in case there are some device mismatches or unexpected channel characteristics during lab testing.

Fig 6.2 shows the simulated waveforms from the testbench. We perform a pulse response test in transient analysis rather than viewing eye diagrams with PRBS input, because the pulse response provides more information about the full system characteristics. The different signals in the figure are interpreted below.

1. **Inputs:** Chapter 3 mentions that the maximum differential peak-to-peak swing at the input of the G_m cells is 320 mV. In the output driver testbench, one G_m cell is de-activated, and the other one is given a 1 UI = 6.25 ps triangular pulse input, with a conservative 300 mV differential peak-to-peak swing and 1 UI rise and fall time, as represented by `v_inp` and `v_inm`.

2. **TX outputs:** The transmitted signals `tx_p` and `tx_m` show some attenuation as expected, since it wasn't possible to achieve a gain ≥ 1 at the driver stage. Some very weak reflections are visible on the TX output pads after a full round trip on the channel. This shows that the termination at the RX input pads is not perfect.

3. **RX inputs:** The received signals `rx_p` and `rx_m` show further attenuation and some ripples after travelling through the channel. No reflections are visible at the RX input pads, which means the reflections got completely attenuated. Hence the termination is good enough for this application.

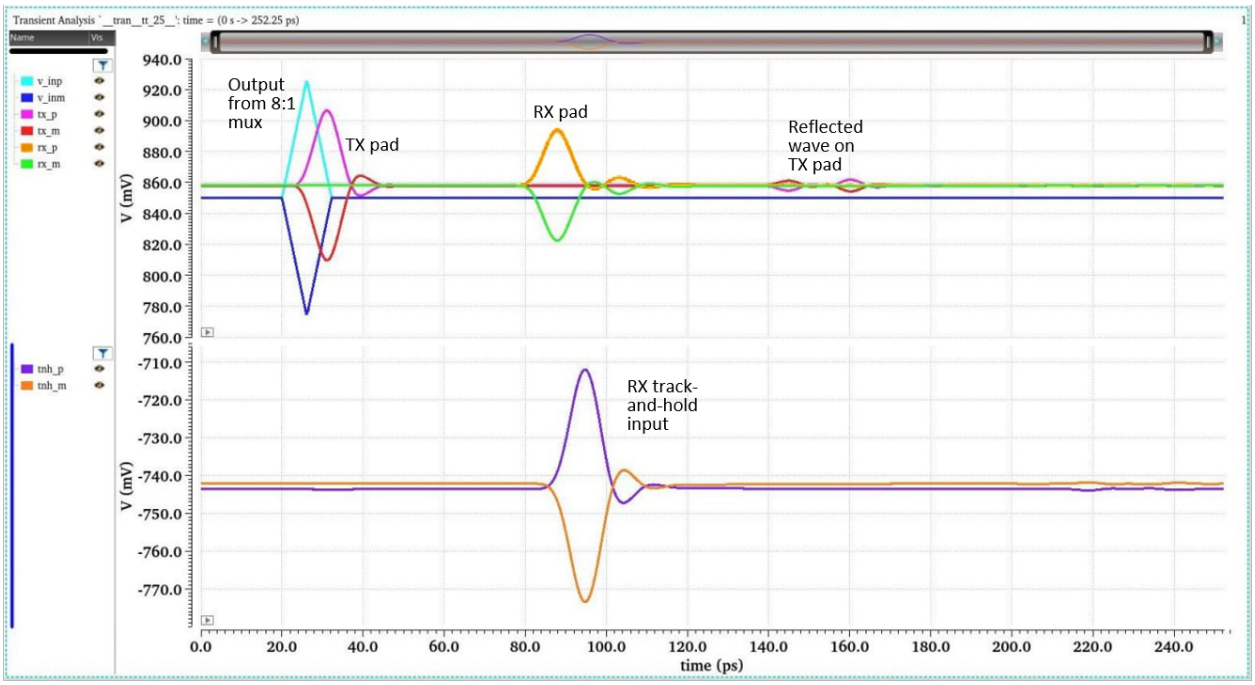


Figure 6.2: Waveforms from Transmitter Output Driver Simulation

4. **T&H inputs:** The common mode voltage at `tnh_p` and `tnh_m` is wrong, and the DC offset between the two nodes is misleading, as expected from the loss of DC bias info in the s parameter representation of the HPF. With the correct DC bias configuration, there is no systematic DC offset between the two inputs of the T&H circuits. The pulse has a differential amplitude of ∼63 mV. This means that differential $h_0 \approx 63$ mV. Based on statistical analysis of the receiver topology, taking into account the gain and noise (voltage + timing) of each stage, residual ISI (inter-symbol interference), DAC offsets, and this T&H input pulse waveform, it is estimated that there is a timing margin of ∼0.5 UI at the target BER = 10^{-12}. This timing margin analysis does not include the effects of phase misalignment, common mode issues, etc. It is assumed that the margin is large enough to accommodate those extra errors.

The output driver stage consumes ∼10 mA current during active operation.

6.2 Complete Transmitter Logic Verification

The logic of the entire transmitter needs to be verified all the way from the PRBS or user specified input upto the outputs at the TX pad in Fig 3.8, to ensure that the correct clock phases were selected for the various intermediate data streams at various stages on the datapath. Chapter 5 mentioned that the clock phase assignment happened at the most critical

clock crossing interface at the input of the high speed 8:1 mux. That assignment determined the phase selection for all the previous mux, retimer, shift register based serializer, and PRBS stages.

While we need to run a transient simulation with the extracted netlist of the entire transmitter datapath for this verification step, we need to be sure that it needs to be run only once because of its prohibitively long run-time. In order to enable quick iterations for debugging the overall TX logic before the final verification, we need to use behavioral models with significantly faster run-times.

Behavioral Modelling Overview with BAG 3++

As briefly discussed in Chapter 1, BAG 3++ can generate a hierarchical behavioral model of the system based on its internal schematic representation. This is advantageous for two major reasons:

1. If a top level non-hierarchical behavioral model is created for a complicated system, there is a risk of missing certain key elements in the model which were introduced in the schematic later during the tapeout cycle. For example, if an odd number of buffers were inserted in the datapath for some reason (delay matching / shortening a critical path for timing closure / etc.), the new output is now an inverted version of the original output. This can be easily missed in the behavioral model, and hence the model and the true circuit go out of sync.

2. By generating a behavioral model directly from the schematic generator, some logic verification simulations can be done much faster in Verilog for iterative debugging purposes. In SystemVerilog models, we can also include the information about analog propagation delay of various cells, to make the latency more realistic and closely matching with schematic or extracted circuit simulations. This feature is taken advantage of in this section to debug any errors in the overall TX datapath logic.

The behavioral modelling in BAG 3++ works as follows:

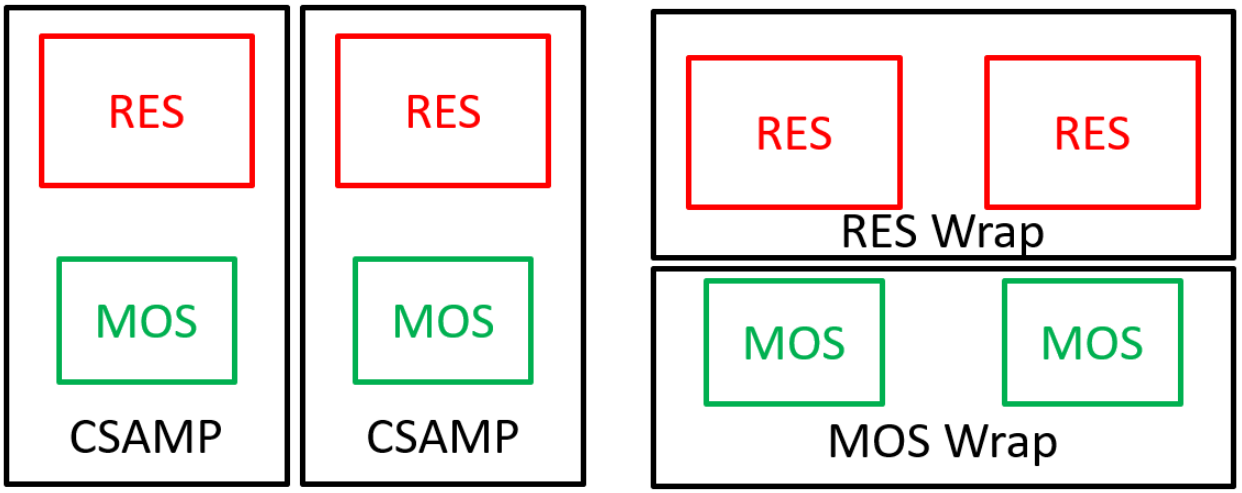
- Users provide the behavioral models of the leaf cells (e.g. digital logic gates, analog cells, etc).

- BAG 3++ constructs the hierarchical behavioral model of the top level cells using the connection information from schematic generators.

- Sequential elements like memory cells (latch, flop) can also be hierarchically assembled by BAG 3++ from the combinatorial models of the leaf cells (tristate inverter, nand, nor, etc).

- Since the behavioral model is generated based on the schematic hierarchy, that also determines the behavioral hierarchy. The layout hierarchy does not necessarily have to

be same, in case a different hierarchy makes more sense from a floorplanning point of view. For example, consider the case of a CS amplifier driving another CS amplifier. From a behavioral modelling point of view, the hierarchy should be a top level with two CS amplifier leaf cells. However, from a layout point of view, it is better to create a 'mos' array with the transistors from both amplifiers, and a separate 'res' array with the resistors from both amplifiers, and assemble those at the top level. This allows for a more compact layout since the transistor and resistor boundary design rules apply only once each around the whole array instead of around each amplifier cell. This is illustrated in Fig 6.3.



(a) Schematic and Modelling Hierarchy

(b) Layout Hierarchy

Figure 6.3: Different Hierarchies for Layout vs Schematic and Behavioral Model

TX Verification: Behavioral Model and Extracted Simulation

The TX datapath logic is debugged and verified using the following steps:

1. First we write the behavioral models of all the leaf cells, annotated with their individual propagation delays, and then generate the hierarchical behavioral model of the entire datapath using BAG 3++.

2. We create a Verilog testbench to run a transient simulation by providing all the necessary clock stimuli and digital calibration signals, with a PRBS input data pattern.

3. Multiple iterations are done to debug errors in the logic, especially the reset synchronization for the shift register based serializers, until the transmitted data pattern perfectly matches the input data pattern. Each Verilog simulation has a run-time of 1 or 2 seconds, allowing us to do multiple rounds of debugging.

4. Now we recreate the exact same transient simulation in a Spectre testbench, with the same stimuli for clock, control signals, and the same input data pattern.

5. We generate the top level TX datapath layout using BAG 3++ and get the parasitic extracted netlist (taking care to replace the magnetic passives with EM extracted s parameter files), and run one long transient simulation. This has a run-time of a few hours to a day or more.

6. We verify that the output data pattern from the Spectre simulation perfectly matches the output data pattern from the Verilog simulation, which in turn matches with the input data pattern. This is the fastest way to debug and verify the overall TX datapath logic.

Fig 6.4 shows the h_0 and h_{-1} data streams before the final combination stage from the Spectre extracted simulation on the top, and the SystemVerilog simulation on the bottom. We clearly see the exact same data pattern in the window enclosed by the black boxes, as well as the 1 UI delay between h_0 and h_{-1}. Note that the voltage swing in the Spectre simulation is worse than what was presented earlier in this thesis. This is because this logic verification step was performed prior to optimizing the high speed 8:1 mux stage and output driver stage to improve the swing.

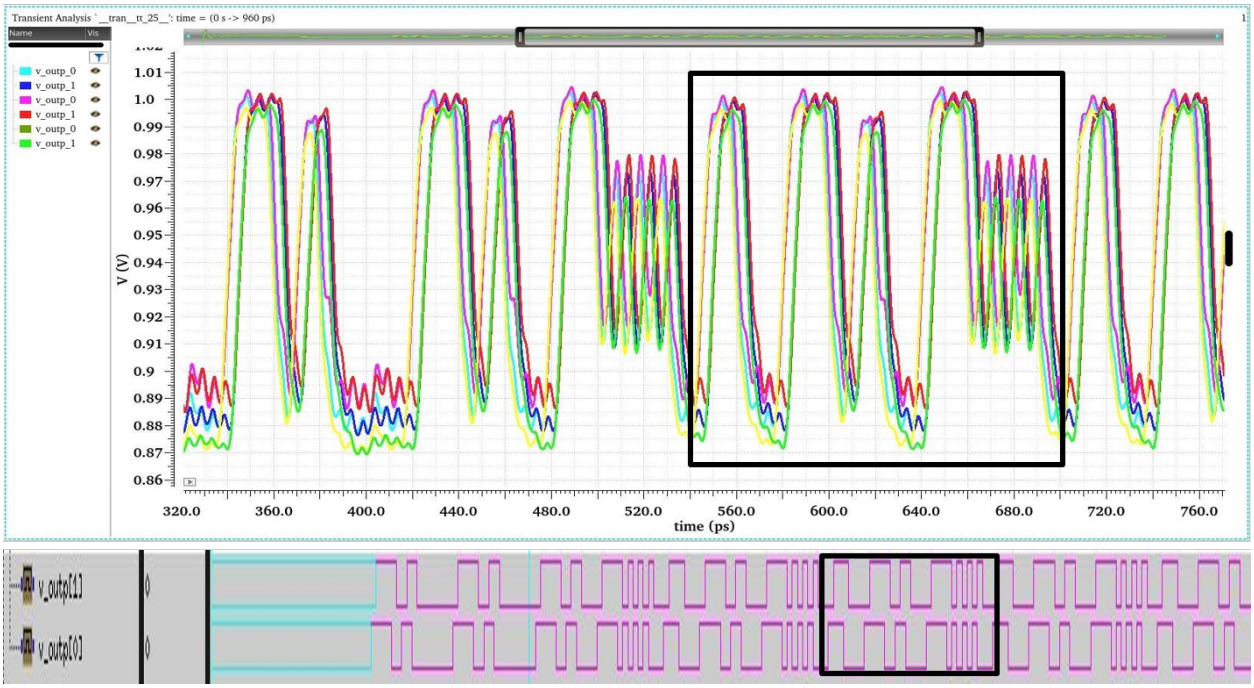


Figure 6.4: TX Logic Verification from Extracted (top) and Behavioral (bottom) Simulations

6.3 Floorplan Iterations using BAG 3++

The use of the BAG 3++ framework played a major role in the design and optimization of the entire 160 Gbps NRZ transceiver project. The layout re-use and automation features saved significant design time and effort across the board, while the automated measurement and design scripts allowed us to perform crucial debugging and verification steps much faster than what would be possible in a traditional AMS flow.

The entire low speed shift register based 128:16 serializer was generated by the same generator scripts that were used in [13]. Generators for many circuit components, like voltage DACs, logic gates, etc. were used in both the transmitter and receiver with appropriate parameters. The clock network was intentionally kept the same in the TX and RX to encourage and allow generator re-use.

In the transmitter datapath design, automated layout generation allowed multiple iterations of the overall floorplan, including placement and routing of the large series and shunt peaking inductors, to optimize the area and performance metrics. Fig 6.5 shows a few wildly different floorplan iterations before we converged on the final design. The change from version 2 to 3 was mainly driven by debugging the TX datapath logic described in the previous section, while the subsequent versions were mainly related to optimizing the design of the peaking inductors.

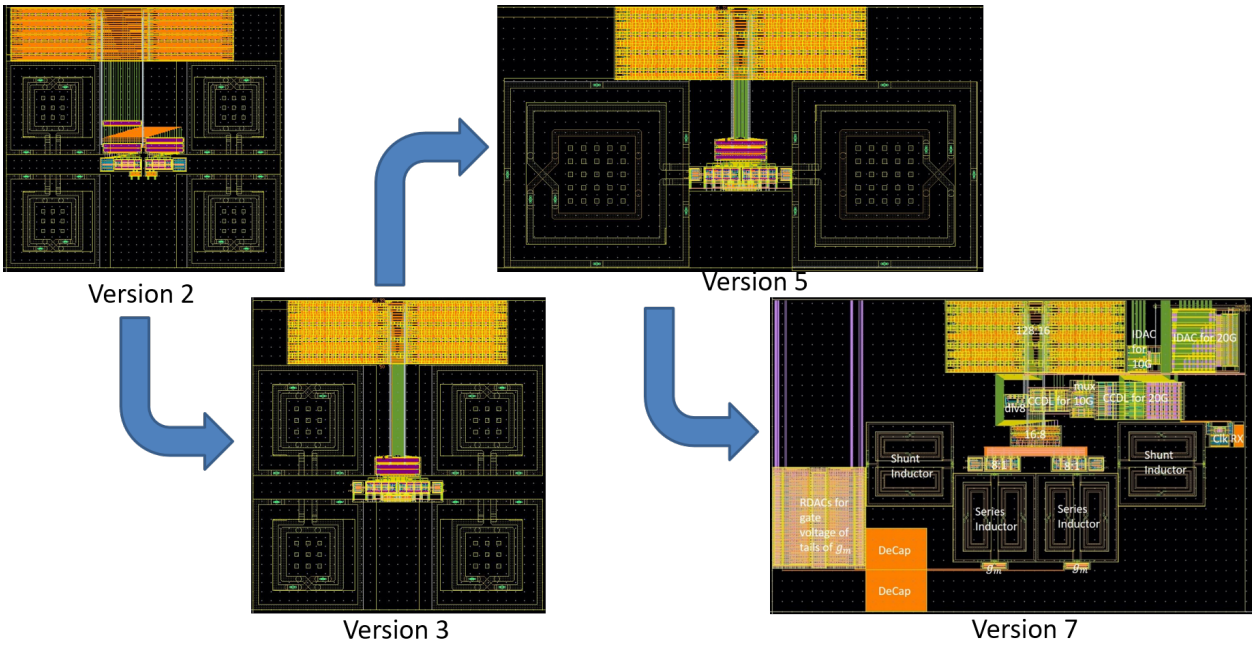


Figure 6.5: TX Datapath Floorplan Iterations in BAG 3++

Chapter 7

Results

7.1 Simulation Results

The energy efficiency from extracted simulations for the 160 Gbps NRZ transmitter is very promising as compared to the two state-of-the-art transmitter designs in literature. Table 7.1 is the updated version of Table 3.1 with the simulated energy efficiency number.

Table 7.1: Comparison of Current Goal with State-of-the-Art Transmitter Architectures

Specs	**JSSC 2022, Kim et al [5]**	**JSSC 2022, Wang et al [13]**	**Current Goal**
Technology node	10 nm FinFET	28 nm planar	16 nm FinFET
Data-rate (Gbps)	224 Gbps PAM4, 112 Gbps NRZ	200 Gbps PAM4, 100 Gbps NRZ	160 Gbps NRZ
Nyquist frequency (GHz)	56	50	80
Equalization technique	8 tap FFE	5 tap FFE	2 tap FFE
Energy Efficiency (pJ / **bit)**	1.88 (with PLL); 1.74 (without PLL)	9.18 (excluding DSP & PLL)	2 (simulated, no PLL)

The power breakdown from simulation results is shown in Table 7.2.
It can be seen that the highest power consuming sub-components in the TX are:

Table 7.2: Transmitter Power Breakdown from Simulation Results

TX Sub-block	**Power Consumption** (mW)
Output driver	10
2× High speed 8:1 mux	10
2:1 mux stage and retimers	30
low speed serializers	70
Clock network	200
Total	**320**

- the clock network, including octature generator, dividers, current controlled delay lines, buffers, and clock distribution network (Fig 3.9)
- the shift register based serializers with local clock buffering.

7.2 PCB Design

In section 3.5, it was mentioned that the 160 Gbps NRZ transceiver will be tested in three modes: loopback, stand-alone TX, and stand-alone RX. Hence the PCB (printed circuit board) needs to be able to support all three modes of testing.

The most stringent requirement for the PCB material comes from the stand-alone RX testing mode. Unlike the stand-alone TX which is connected to probe pads on the package, the stand-alone RX has its inputs connected to solder balls. This means the external signal generator has to feed data signals onto the PCB, which then get routed to the package. Hence the PCB prepreg dielectric should have very low dissipation factor (Df) to avoid signal loss at high frequencies. The most commonly used cheap PCB dielectric is FR4 (flame retardant 4), which has a high Df of 0.02 that is not suitable for our application. The dielectric with the lowest Df available from the PCB manufacturing company Sierra Circuits was Astra MT (MegTron) 77 with a Df of 0.0015, which is what we opted for. The recommended 6 layer stack-up from Sierra is shown in Fig 7.1. Only the top and bottom prepreg use the MT77 dielectric while the core dielectric is FR4 (FR-370HR is Sierra's code number for FR4).

We decided to divide the PCB into two types of boards, namely the 'main' board and the 'auxiliary' board. The boards were manufactured by Sierra Circuits, and then all the components were assembled by DigiCom Electronics.

1. **Main:** This PCB houses the triple die attached package (Fig 3.13), the supply regulation collateral, and the connectors for external clock and data inputs.
2. **Aux:** This PCB houses the level shifters for communicating with the FPGA.

Imp	Lyr	Cu Usage	Type	Image	Foil	Thk (Mil)	Plt (Mil)	Er	Family	Generic Name
	sm1					0.5				
1xΩ	L1	24 / 64	Signal		0.5oz	5	0.8	3.00	MT77	0.005 HHxHH
	L2		Power / Ground		0.5oz		0	3.63	FR-370HR	1080
						6.696		3.63	FR-370HR	1080
	L3	64 / 64	Power / Ground		0.5oz	31	0	4.24	FR-370HR	0.031 HHxHH
	L4		Power / Ground		0.5oz		0	3.63	FR-370HR	1080
						6.696		3.63	FR-370HR	1080
1xΩ	L5	64 / 24	Power / Ground		0.5oz	5	0	3.00	MT77	0.005 HHxHH
	L6		Signal		0.5oz		0.8			
	sm6					0.5				

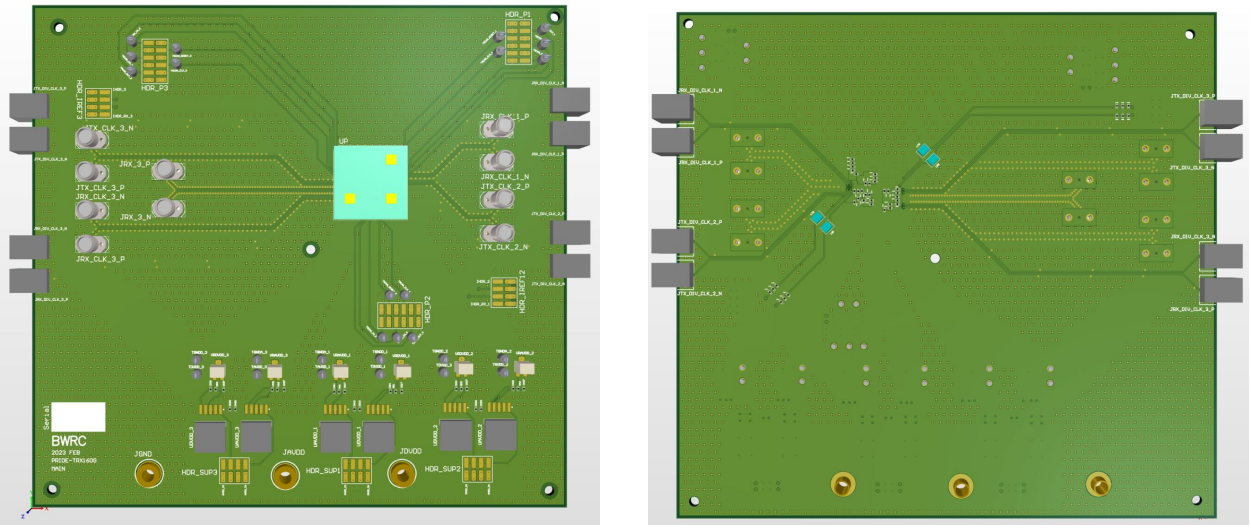
Impedance Table

Index	Layer	Required Impedance [ohms]	Tol [ohms] +/-	Type	Ref 1	Lower Ref	Orig LW [mil]	Orig Spacing [mil]	Orig CP Spacing [mil]	Finished LW [mil]	Finished Spacing [mil]	Fin. CP Spacing [mil]	Impedance Simulation [ohms]	Impedance before Mask [ohms]	IMG
a	1	50	10%	CP SE Uncoated		2	10.50		6.00	10.50		6.00	49.7		
b	6	50	10%	CP SE Uncoated		5	10.50		6.00	10.50		6.00	49.7		

Figure 7.1: Recommended 6 layer PCB stack-up

Main PCB Design

We wish to re-use each package for all three testing modes, hence the main PCB has connectors and routing to enable all of them. The front and back views of the main PCB are shown in Fig 7.2 and the various components are elaborated below:



(a) Front view (b) Back view

Figure 7.2: 3D views of Main PCB from Altium Designer

1. The cyan square in the front view is the triple die attached package. The yellow squares

in the cyan square denote the locations of the 3 dies. Based on Fig 3.13, it is clear that die 3 is on the west, and dies 1 and 2 are on the east.

2. Since the TX on die 2 and the RX on die 1 are used for the loopback mode test, the relevant connectors are placed on the east side in the front view.
    - four 'K' (2.92 mm) connectors for external differential 20 GHz clock inputs to the TX and RX
    - four edge SMA connectors for viewing the divided 1.25 GHz differential clock outputs from the TX and RX
    - two 2×6 headers for bringing in the level shifted FPGA signals for the TX and RX from the auxiliary PCB into the main PCB
    - one 2×4 header for bringing in external bias current from SMUs (source measure unit) for TX and RX
3. Since the TX and RX on die 3 are used for stand-alone testing modes, the relevant connectors are placed on the west side in the front view.
    - two 'W' (1.85 mm) connectors for external differential data input to the RX
    - four 'K' (2.92 mm) connectors for external differential 20 GHz clock inputs to the TX and RX
    - four edge SMA connectors for viewing the divided 1.25 GHz differential clock outputs from the TX and RX
    - one 2×6 header for bringing in the level shifted FPGA signals for both TX and RX from the auxiliary PCB into the main PCB
    - one 2×4 header for bringing in external bias current from SMUs (source measure unit) for TX and RX
4. The north side in the front view of the min PCB is kept clear so that the triangular probe head can be brought in without any collisions or obstructions to land on the probe pads for the outputs of the TX on die 3.
5. The south side in the front view on the PCB has LDOs, potentiometers and supply decap for each of the six supplies on the package (AVDD and DVDD for each of the three dies). The power regulation needs to be on the main PCB to ensure clean supplies for the dies. If power regulation were on the auxiliary PCB instead, it would need to be cleaned again after landing on the main PCB.
6. The back view of the main PCB shows the local supply decap for all 6 power domains placed directly under the package. There are also inductors for improving the input impedance of the external bias currents.

During lab testing, we realized that the main PCB size was too large which made it impossible to properly place under the microscope at the probe station, so some components had to be depopulated from some of the main PCBs to allow for a good fit. In hindsight, creating two different variants of the main PCB, with one variant supporting the loopback and the stand-alone RX testing, and the second variant supporting the stand-alone TX probing, would have made the PCBs easier to design and cheaper to manufacture as the area could be significantly reduced. The major motivation to have one combined main PCB was to re-use each package for all 3 testing modes, but we ended up hacking some main PCBs anyway for probing.

Auxiliary PCB Design

Fig 7.3 shows the front view of the much simpler auxiliary (aux) PCB. There are no components on the back-side. This boards needs to support only scan signals which are at 1 MHz or lower, so the low Df Astra MT77 dielectric is not necessary for this PCB. Nevertheless, since the large panels used to manufacture the main PCBs had left-over unused areas, Sierra Circuits recommended using the same stack-up for the aux PCB.

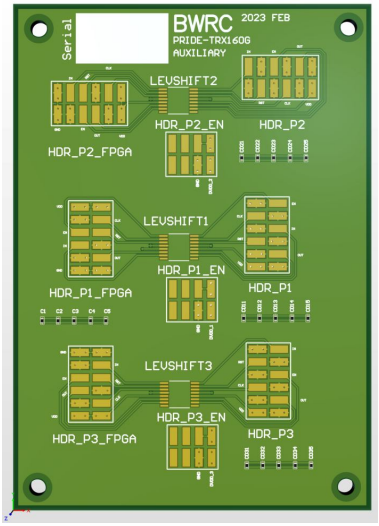


Figure 7.3: 3D view of Auxiliary PCB from Altium Designer

Each of the three dies on the package require the following set of components on the aux PCB, in addition to supply decap network:

- one 2×6 header for receiving the scan signals from the FPGA
- one level shifter to convert the 3.3 V domain FPGA signals to the regulated DVDD domain on the main PCB

- one 2×6 header for send the level shifted scan signals to the main PCB
- one 2×4 header for the 'enable' signal of the level shifter

7.3 Lab Testing Setup & Measurement Results

For the FPGA (field programmable gate array), we chose the OpalKelly XEM 7001. It comes with an easy-to-use Python API for programming the various necessary functions for the transceiver setup and testing.

There were two main testing setups in the lab, relevant to the transmitter:

1. **Probing Setup:** This is shown in Fig 7.4, where the red arrows represent the ribbon connections between the FPGA, aux PCB, and main PCB, and the blue triangle represents the probe head. Note that the main PCB in this figure has some components depopulated for fitting under the microscope in the probe station. Unfortunately we could not get any results from this setup since the computer at the BWRC probe station stopped working, and we could not load the software for operating the probe station components.

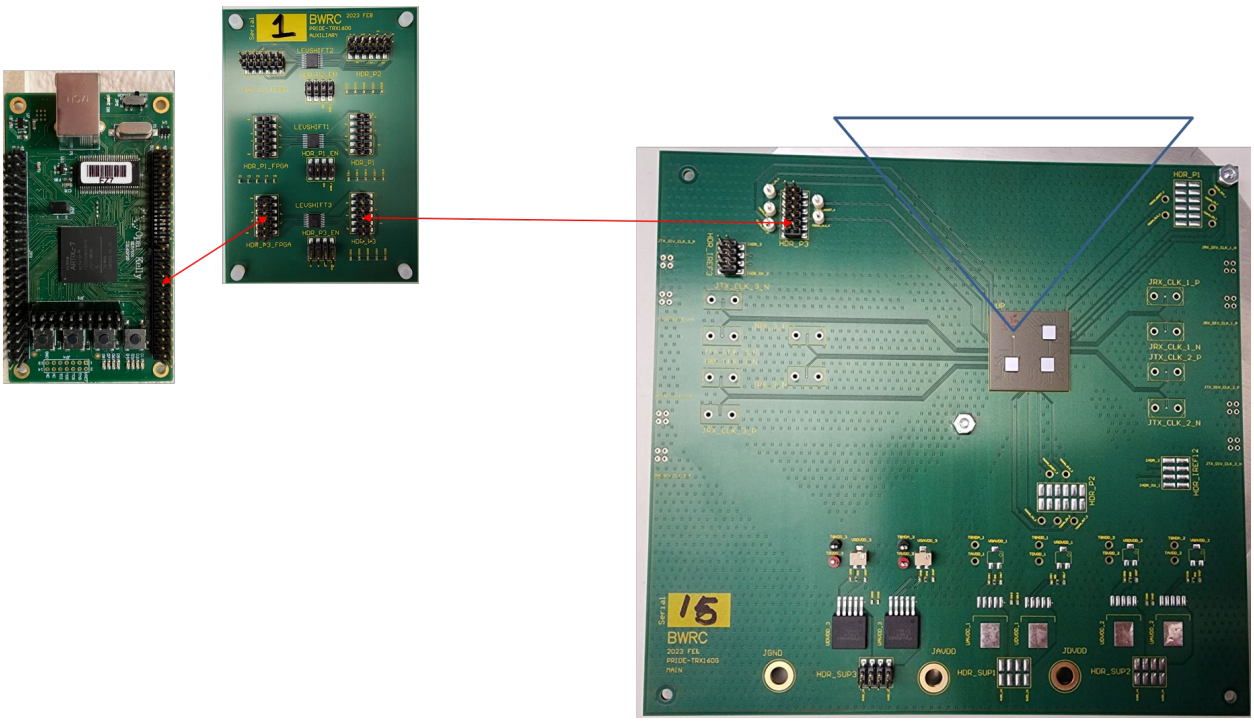


Figure 7.4: Setup of PCBs and FPGA for stand-alone TX probing

2. **Loopback Setup:** This is shown in Fig 7.5, where the red arrows represent the ribbon connections between the FPGA, aux PCB, and main PCB.

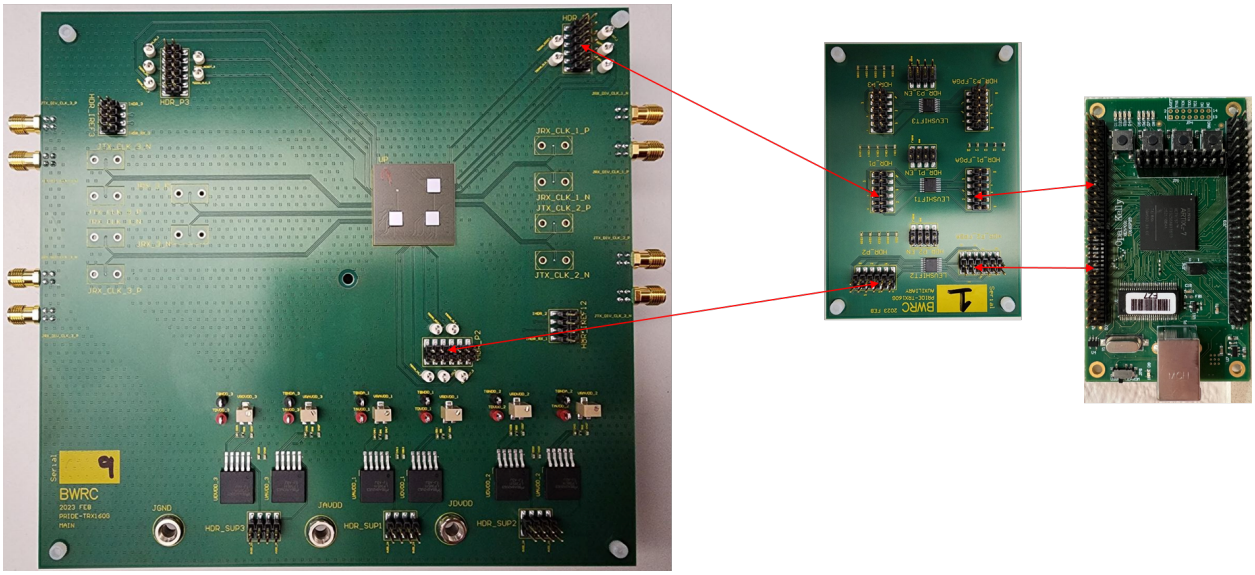


Figure 7.5: Setup of PCBs and FPGA for loopback testing

This setup provided the following results:

a) The scan chain was verified to be fully functional after power-up.

b) Current consumption from AVDD and DVDD matched the expected numbers from extracted simulations, in both idle and active modes.

c) The clock network on the transmitter side is verified to be working by looking at the output divided clock waveform. Using the manual calibration knobs, we varied the quadrature 10 GHz clock phase that is chosen to create the divided clock, to verify that all the clock phases are alive.

- In the free running situation (no external differential 20 GHz clock injection), the octature phase generator locks to 22.4 GHz, which is apparent from the fact that the divided clock frequency is 1.4 GHz in Fig 7.6.
- With external differential 20 GHz clock injection, the divided clock frequency is exactly 1.25 GHz in Fig 7.7, as expected from the design.
- With external differential 10 GHz clock injection, the divided clock frequency is almost exactly 625 MHz in Fig 7.8, as expected from the division ratio. This case was a pleasant surprise since we didn't expect that the octature generator would be able to lock to half of its free running frequency. Indeed, we can see that the divided clock waveform is not a perfect sinusoid in this case and has some higher order harmonics.

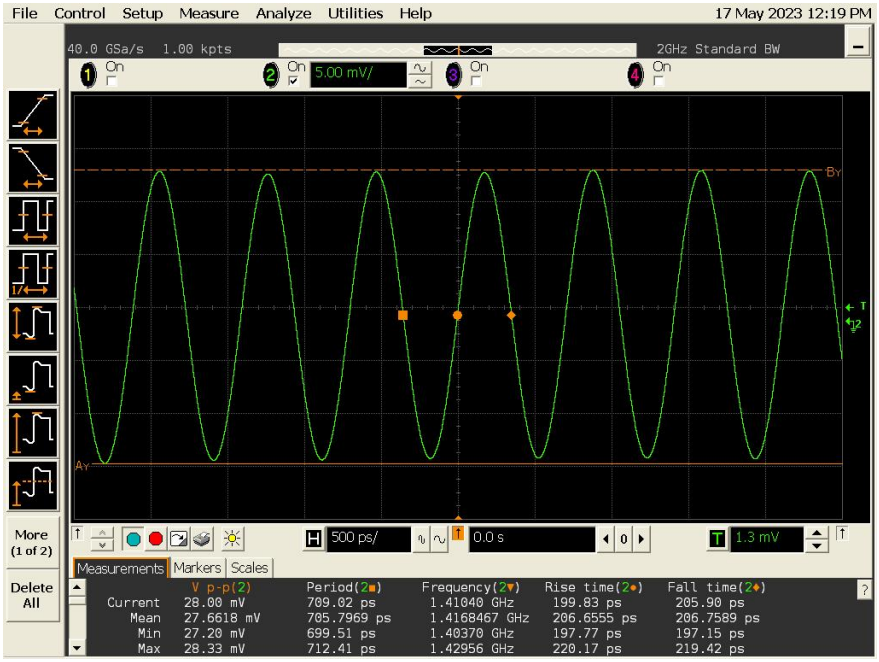


Figure 7.6: Output Divided Clock (Free Running)

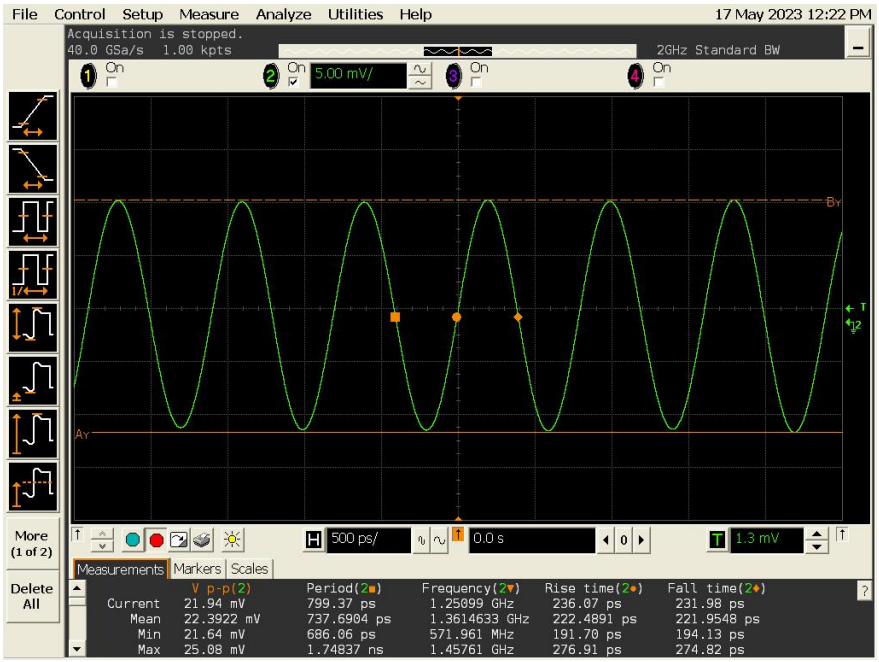


Figure 7.7: Output Divided Clock (20 GHz Injection)

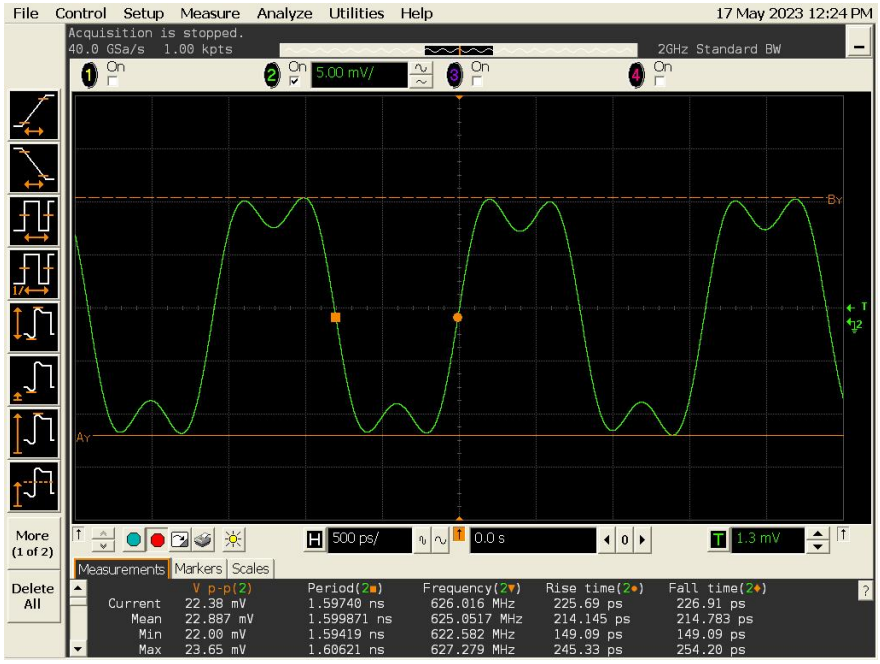


Figure 7.8: Output Divided Clock (10 GHz Injection)

d) Unfortunately we could not make any further progress in this test setup, because the clock network on the receiver had severe phase mismatch issues that made the receiver datapath essentially inoperable at the target frequency. Hence the loopback test could not be performed to evaluate the transmitter datapath performance.

Chapter 8

Conclusion

8.1 Thesis Contribution

The major focus of my PhD research has been the automation of AMS (Analog & Mixed Signal) design using generators and design scripts, with particular focus on ultra high speed wireline design. To that end, this thesis mentions substantial contributions to the development and maintenance of the Berkeley Analog Generator (BAG) framework, as well as efforts to push the boundary of achievable data rate and energy efficiency in wireline communication systems.

A significant portion of my early PhD years were focused on ramping up on BAG. My unquenchable curiosity to always find out how anything 'really' works, meant that I would never be satisfied with just being a BAG user who only deals with the user facing API to create generators and design scripts, and remains oblivious to the back-end implementation. I absolutely *needed* to know the core principles and implementation details of every single feature, so that I can then add newer features, improve on the existing framework, debug issues that many users were often content with working around, and even challenge and modify some outdated conventions which didn't make logical sense to me.

With the great power of being a BAG developer, came the great responsibility of setting up and maintaining BAG workspaces in a wide variety of technology nodes, ranging all the way from sub µm planar nodes to nm FinFET nodes, from multiple different foundries. Having access to all these technologies gave me exposure to a wide variety of design rules and device characteristics, and ultimately made me a better AMS designer as I learned the core design methodologies for AMS circuits that can be transferred across different PDKs. It also gave me the opportunity to interact with and get feedback from multiple researchers working on different projects, which helped to not only improve the BAG framework but also added to my own circuit design knowledge.

Of course, a new AMS design framework that challenges the traditional design flow, will be accepted and adopted widely only if we can design IP blocks that have similar or better performance than existing designs, with lower manual design effort. To that end, my research

team targeted the design of the ultra high speed 160 Gbps NRZ transceiver, which is faster than any wireline communication system that has ever been demonstrated.

Due to unfortunate supply-chain delays with dies, package and PCB manufacturing that interfered with our PhD timelines, and the eventual equipment failure in the lab, we didn't succeed in demonstrating a fully working transceiver. However, we did get some promising results, learned many valuable lessons, and even helped Intel harden their 16 nm FinFET process through years of feedback and collaboration.

Some critical take-aways from the transmitter datapath design are summarized below:

- It would have been preferable to use different analog supply domains. A nominal 1 V supply domain for the CMOS stages (to avoid exceeding gate breakdown voltages during full swing operation), and a separate higher (e.g. 1.5 V) supply domain for the high speed CML stages, like the 8:1 mux or the output driver, would allow us to get larger gain-bandwidth and voltage swings at the frontend. Unfortunately the package design and bump map were fixed about a year prior to the design optimization efforts, so it wasn't possible to accommodate more supply domains when the need arose.

- Design of magnetic peaking structures is extremely challenging, especially in advanced technology nodes, because of the higher order parasitic effects that are not accounted for in theoretical analysis. While inductive peaking still remains a very promising and useful bandwidth extension technique, converging to the optimal design is a very difficult task that should be handled by some automated optimization framework, as highlighted in [4]. This aspect of the transmitter datapath design gave birth to an entirely new feature in BAG 3++: the electromagnetic simulation support. Significant time and efforts were spent on implementing the `EmSimAccess` class and the advanced `OptDesigner` with EM just for the purpose of designing the high speed 8:1 mux stage.

- While many semesters of efforts went into the design of the transceiver, a disproportionately small amount of time and effort was put into planning for lab testing. Some of the issues we faced in the lab, like the shortcomings that highlighted the benefit of separate PCBs for the probing vs no-probing test modes, could have been alleviated if we had planned ahead better from the very beginning of the project.

Despite the issues, the thesis still makes a very meaningful contribution to the field of wireline communications research, as well as design automation. It explores and challenges the limits of high data rates feasible in a technology node. It shows that automation frameworks can be used to design high performance energy efficient circuits. As elaborated in Chapters 4, 5, 6, various aspects of the BAG 3++ framework, like closed loop design scripts, behavioral modelling, etc, were instrumental in the design and verification process. Generators and measurement scripts for various sub-blocks developed for this project are already being re-used by other researchers in newer SerDes and data converter projects at BWRC.

8.2 Future of Wireline SerDes Transmitters

The main reason why serial links became popular over parallel links back in the 80's and 90's was the efficiency of transferring data across a single wire instead of a bus of wires. This drove serial links research to achieve higher and higher data-rates over the years, which was also helped by technology scaling benefits. However, with device scaling finally slowing down at the single digit nm nodes, it looks like the gain bandwidth limit of technology nodes has finally been reached at the AMS front-end circuits, making it difficult or infeasible to increase the single lane data-rate any further. Higher order modulation schemes like PAM-4 instead of NRZ increase the spectral efficiency at the expense of lower SNR, stringent linearity constraints, and more complicated equalization schemes. That is why parallel links are seeing a resurgence in popularity, with AIB (Advanced Interface Bus), UCIe (Universal Chiplet Interconnect Express), BoW (Bunch of Wires) protocols becoming more mainstream. Even though these interfaces use a parallel bus of wires with lower data rate on individual lanes, advanced packaging techniques allow extremely dense packing of the interfaces at tight bump pitches, which makes the aggregate shoreline bandwidth density higher than serial links. The parallel links also use single ended signalling instead of differential signalling in standard SerDes, which can reduce the power consumption by almost 50%. Hence, most wireline communications research will probably shift to parallel links more and more in the coming years, and / or explore optical I/O solutions.

8.3 Future of Automated Generators and Closed-Loop Design Scripts

Automation of layout generation and circuit measurements is causing a paradigm shift in the traditional AMS design industry. While digital PnR flows have been automated for many years, researchers are finally getting promising breakthroughs in AMS automation now. Different variants and spin-offs of BAG have already been adopted to various extents in industries and universities around the world along with their own in-house automation tools, e.g. the MOSAIC (Modular Open Source Analog IC Design) collaboration. Especially with the recent AI boom, various research teams are already exploring AMS design optimization using AI. Thus it is absolutely essential for AMS designers to start adopting generator based design methodologies immediately, or face the risk of falling behind in today's fast-paced industry. Automated generators and closed loop design scripts are definitely here to stay and advance further over the coming years, highlighting the relevance of this thesis.

Bibliography

[1] Eric Chang et al. "BAG2: A process-portable framework for generator-based AMS circuit design". In: *IEEE Custom Integrated Circuits Conference (CICC)*. San Diego, CA, USA, 2018, pp. 1–8.

[2] Ming-Shuan Chen and Chih-Kong Ken Yang. "A 50-65 Gb/s Serializing Transmitter With a 4-tap, LC Ladder-Filter-Based FFE in 65-nm CMOS". In: *IEEE Journal of Solid States Circuits (JSSC)* 50.8 (Aug. 2015), pp. 1903–1916.

[3] Denis C. Daly, Laura C. Fujino, and Kenneth C. Smith. "Through the Looking Glass - 2020 Edition". In: *IEEE Solid-States Circuits Magazine* (Jan. 2020), pp. 8–24.

[4] Jaeha Kim et al. "Design Optimization of On-Chip Inductive Peaking Structures for 0.13-μm CMOS 40-Gb/s Transmitter Circuits". In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 56.12 (Dec. 2009), pp. 2544–2555.

[5] Jihwan Kim et al. "A 224-Gb/s DAC-Based PAM-4 Quarter-Rate Transmitter With 8-Tap FFE in 10-nm FinFET". In: *IEEE Journal of Solid States Circuits (JSSC)* 57.1 (Jan. 2022), pp. 6–20.

[6] Greg LaCaille et al. "Design and Demonstration of a Scalable Massive MIMO Uplink at E-Band". In: *IEEE International Conference on Communications Workshops*. Dublin, Ireland, June 2020.

[7] Thomas H. Lee. *The Design of CMOS Radio-Frequency Integrated Circuits*. Cambridge University Press, Dec. 2003.

[8] Jeyanandh Paramesh and David J. Allstot. "Analysis of the Bridged T-Coil Circuit Using the Extra-Element Theorem". In: *IEEE Transactions on Circuits and Systems II: Express Briefs* 53.12 (Dec. 2006), pp. 1408–1412.

[9] Behzad Razavi. "The Bridged T-Coil". In: *IEEE Solid-States Circuits Magazine* (Feb. 2015), pp. 9–13.

[10] Behzad Razavi. "The Design of Broadband I/O Circuits". In: *IEEE Solid-States Circuits Magazine* (June 2021), pp. 6–15.

[11] S. C. Dutta Roy. "Comments on Analysis of the Bridged T-Coil Circuit Using the Extra-Element Theorem". In: *IEEE Transactions on Circuits and Systems II: Express Briefs* 54.8 (July 2007), pp. 673–674.

[12] Zhongkai Wang et al. “An Automated and Process-Portable Generator for Phase Locked Loop”. In: *58th ACM/IEEE Design Automation Conference (DAC)*. San Francisco, CA, USA, 2021, pp. 511–516.

[13] Zhongkai Wang et al. “An Output Bandwidth Optimized 200-Gb/s PAM-4 100-Gb/s NRZ Transmitter With 5-Tap FFE in 28-nm CMOS”. In: *IEEE Journal of Solid States Circuits (JSSC)* 57.1 (Jan. 2022), pp. 21–31.