Design of an Analog Receiver Frontend Implementing the USB 2.0 Spec *

Ansh Chaurasia
Electrical Engineering and Computer Sciences, EECS
University of California, Berkeley
Berkeley, United States of America
achaurasia@berkeley.edu

***Abstract*—In this paper, we discuss the design and implementation of a receiver facing analog frontend driver (AFE) module that is part of a broader EECS 251B USB 2.0 implementation. State of the art AFE modules for the latest USB spec - USB 4 at this paper's time - face common problems of creating differential receivers designed for high bandwidth and minimal jitter. Our USB-2 AFE differential receiver, powered by an implementation of the strongARM comparator latch, supports bandwidth speeds of 1.5 MB/s, 12 MB/s, and 480 MB/s. The highest bandwidth mode presents interesting analog design challenges in designing a driver capable of discerning low voltage differences, which we derive results for using Cadence Virtuoso simulations. Overall, we present the AFE design with simulation data, a layout implementation using Sky130's PDK that is DRC and LVS clean, and integrated with additional parts of the USB chip to comprise a broader implementation.**

***Index Terms*—USB 2.0, analog design, analog frontend, differential receiver**

I. Background

The Universal Serial Bus (USB) protocol is a widely adopted standard for short-distance, serial digital data communication. It's truly become a ubiquitous protocol. The majority of computer devices and a stunning 68% of new smartphone models released by companies such as Samsung, Huawei, LG, and Xiaomi are compatible with charging mechanisms using USB-C, a variant of the protocol. USB itself provides a standard interface for data transfer, power delivery, and device control, with support for plug-and-play and hot-swapping capabilities. In this work, we will design devices implementing the USB 2.0 spec, enabling data rates as high as 480 MB/s.

II. Problem - Differential Data Retrieval

USB 2.0 implements data transfers through a complementary differential pair of data lines $D+$ / $D-$, and requires two sets of these signals to (1) transmit the data to the *host device* (typically a computer), and (2) receive data from the host. The AFE Design in question here focuses on providing an comparator interface from **the data lines** to **internal USB-2 modules** doing additional processing.

A. The RX Frontend Top-Level Interface

As suggested by Figure 1, our AFE will directly consume data from the two differential channels $D+$ / $D-$ and, outputting a series of RX signals used by the remainder of the USB-2 implementation.

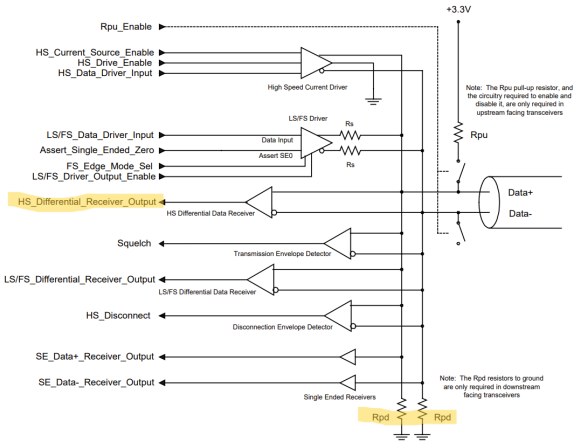

Fig. 1. Reference implementation provided by the USB 2.0 spec, Figure 7-1 that we intended to follow. See Figure 2 for the actual spec post modifications. The scope of our block is highlighted in yellow. Our module was designed to consume differential outputs from $D+$ and $D-$ input signals, and output differential signal *HS Differential Receiver Output*.

III. Summary of Specs Derived

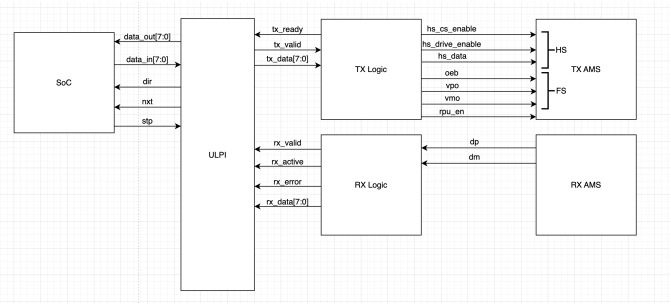

Fig. 2. The scope requested by downstream processing blocks consuming differential signals. Our role is to implement the RX AMS block, and we provide two signals instead of the expected HS Differential Receiver output.

This project was done in conjunction with other teams developing various components of the USB specification. A secondary consumer of our block focusing on data recovery requested we provide *both differential receiver output signals*, $D+, out$ and $D-, out$, which they could utilize for downstream processing.

IV. Preliminary Design Selection

A. Comparator Selection

Early USB 2.0 HS (480 Mb/s) transceiver designs commonly employed *current-mode differential comparators* optimized with window-enabling logic and analog common-mode feedback to meet the rigorous speed and common-mode resilience requirements of the USB 2.0 specification (480 Mb/s, VHSCM = $-50mV$ to $500mV$) [1]. Specifically, Li *et al.* presented a UTMI-compatible physical-layer USB 2.0 transceiver in 0.13 µm CMOS that used such a comparator topology to achieve $BER < 10^{-12}$ while satisfying technology-specific constraints [1].

Subsequent works leveraged the *StrongARM latch comparator* for its zero static power dissipation and rail-to-rail output swing—a combination particularly attractive for low-power host and device transceivers. Razavi's comprehensive survey documents the StrongARM topology's prevalence in wireline and mixed-signal front-ends, highlighting its high sensitivity and fast regenerative action [2].

Beyond topology choices, *device sizing strategies* also play a critical role. Chen's dissertation shows that upsizing the precharge devices in a StrongARM latch can reduce input-referred thermal noise by up to 30% for a given energy budget, an insight applicable to USB front-ends requiring high sensitivity under process and temperature variations [3]. Other works have applied offset-cancellation techniques in latch comparators to mitigate device mismatch without sacrificing speed, using dynamic calibration loops that are compatible with the USB 2.0 full-speed and high-speed common-mode windows [4]. Finally, several industrial USB 2.0 HS transceiver ICs (e.g., TI's TUSB1210 OTG transceiver) are documented to employ tuned StrongARM or current-mode comparator variants to meet both the 45±10% input impedance and the VIL/VIH threshold bounds listed on spec [5].

B. Why we converged to the StrongARM comparator design

Given its widespread use, noise reduction potential, and relative simplicity, we opted to proceed with a simplified version of the strongARM comparator used in industry applications.

C. StrongARM Comparator Fundamentals

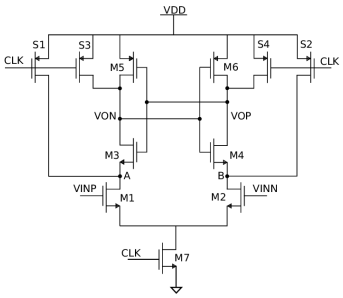

Fig. 3. Diagram of the strongARM comparator referenced throughout the design and layout process.

The StrongARM comparator is a fully dynamic, latched differential comparator widely used in high-speed, low-power analog-to-digital converters (ADCs). It operates in two distinct phases—*reset* and *evaluation*—controlled by a clock signal ϕ.

a) Reset Phase ($\phi = 0$):: During the reset phase, internal nodes are precharged to supply or ground, typically:

$$\text{Out}_p = \text{Out}_n = V_{DD}, \quad V_X = V_Y = 0$$

where V_X and V_Y are internal nodes connected to the drains of the differential input pair.

b) Evaluation Phase ($\phi = 1$):: When the clock goes high, the tail NMOS transistor turns on, allowing the differential input pair to steer current based on the differential input voltage $\Delta V_{\text{in}} = V_{in+} - V_{in-}$.

V. Schematic and Design Choices

In this section, we focus on the design of the strongARM latch, key design variables, and our attempts towards creating a comparator with minimal propagation time, and sensitivity consistent with the USB 2.0 spec. Before any further discussion, we cite the USB 2.0 requirements for the given block.

A. USB-2.0 Specification Constraints

The spec provides the following options for amplifier design - *It is left to transceiver designers to choose between incorporating separate high speed and low-/full-speed receivers, as shown in Figure 7-1, or combining both functions into a single receiver.*

We opt to create one single differential amplifier, sharing constraints from the low, full, and high speed receiver. The constraints we consider from the USB 2.0 spec are listed below for convenience.

1) **Minimum, Input Side Impedance**: The DC resistance from D+ or D- to the device ground is required to be *45 ±10%* when measured without a load
2) **Common Mode Voltage Resilience**: A high-speed receiver should be able to reliably receive such signals in the presence of a common mode voltage component (VHSCM) over the range of –50 mV to 500 mV (the nominal common mode component of high-speed signaling is 200 mV).

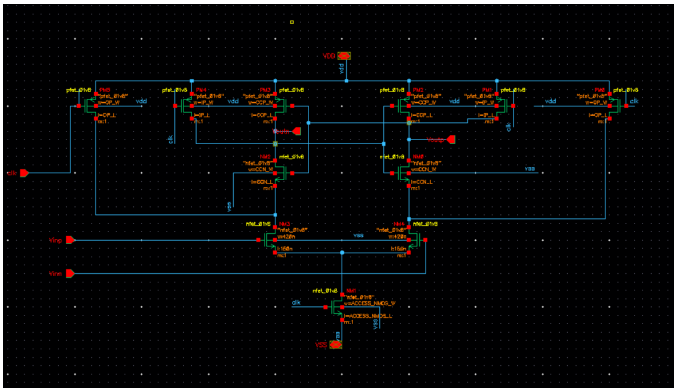

Fig. 4. Schematic used to perform high level simulations. Cadence's Virtuoso software was utilized for these ends.

B. StrongARM Comparator Design

The most crucial portion of the differential receiver is the design of the strongARM latch. Above shows the final Cadence Virtuoso schematic for the strongARM latch, and the critical design variables are listed below:

- *General Transistor Length*: Length for all transistors from the PDK we use (Sky130, nfet_01v8 + pfet_01v8)
- *Access NMOS Transistor Width*: Set to a default of $420n$ for the simulation purposes
- *Cross Coupled NMOS / PMOS Transistor Width*: Set at $420n$ and $550n$ respectively.
- *Inner / Out Precharge PMOS Transistor Width*: Set at $550n$ and $550n$ respectively.

Transistor sizing plays a critical role in balancing speed, noise resilience, and input impedance—key requirements for meeting USB 2.0 compliance across low, full, and high-speed modes. Wider transistors in the differential pair (cross-coupled NMOS/PMOS) enhance transconductance (g_m), which improves sensitivity and decision speed but at the cost of increased input capacitance and potential kickback noise.

Conversely, the access NMOS and precharge PMOS widths impact evaluation speed and regeneration time. A common sizing strategy is to begin with minimum channel lengths for speed, and size widths to ensure sufficient gain and matching, while managing trade-offs with capacitive loading and static input resistance. In our design, transistor widths were selected to achieve a balance between meeting the 45 Ω input resistance requirement, maintaining the required switching threshold range, and ensuring robust operation within the specified common-mode voltage window.

C. Simulations and Verification of the USB Spec

First, we share output waveforms from the verification setup to demonstrate the working functionality of the strongARM comparator in various cases. Additionally, we share the testbench schematics for maximum transparency.

To test pure functionality for our schematic, we ran simulations at $f_{clk} = 1$ KHz, verifying that the output of our

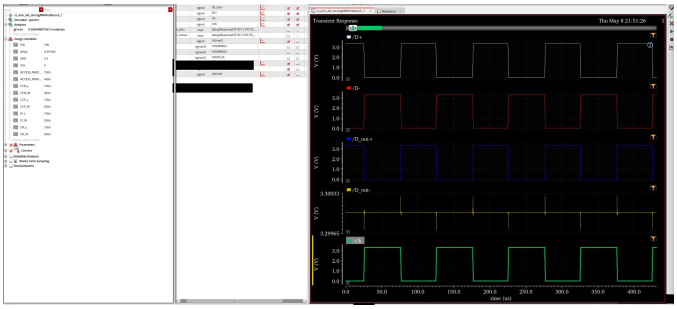

Fig. 5. Virtuoso Maestro Simulation of the strongARM testbench. We provide the clk pulse signal, and $D+$ and $D-$ as inputs. See test schematic for further details.

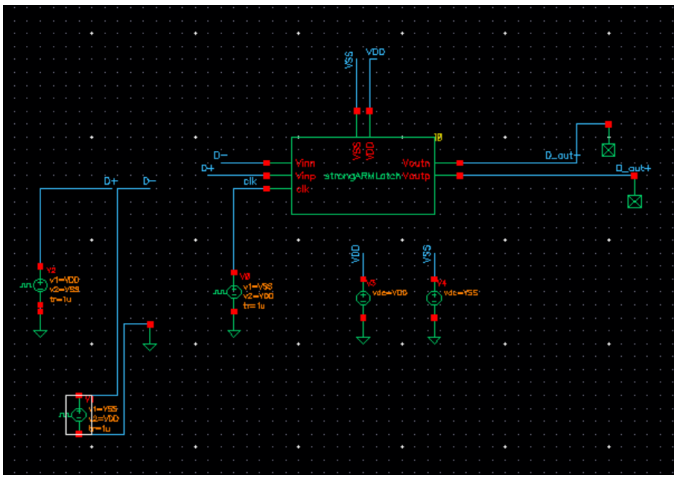

Fig. 6. General testbench schematic setup to evaluate the functionality of the strongARM comparator.

simulations was as expected. More specifically, we defined differential signals $D+$ and $D-$ in the testbench, and flipped whether they started with initial value V_{DD} or V_{SS}. Both of these pulse signals were in phase with the clock signal, meaning that on a clock high, each differential signal has a value of V_{DD} or V_{SS}. We verified that the output differential signals behaved as follows, as per strongARM functionality:

- During the reset / regeneration phase, both differential output signals $D_{out,+}$ and $D_{out,-}$ rose to the analog V_{DD} value of 3.3.
- At the rising edge of the clock, we see "lower" output signal falling in voltage to V_{SS}. In the reference testcase Figure 6, the $D_{out,+}$ falls as intended since $D- > D+$.

At the 1 KHz clock frequency, we were able to successfully verify the following behaviors:

- $D- > D+$ leads $D_{out}, +$ falling to V_{SS}, while $D_{out,-}$ remains at V_{DD}
- $D+ > D-$ leads $D_{out}, -$ falling to V_{SS}, while $D_{out,+}$ remains at V_{DD}
- $D_+ = D_- = V_{DD}$ leads both differential output signals to plateau to V_{SS}.

- $D_+ = D_- = V_{SS}$ leads both differential output signals remaining at V_{DD}.

1) Expectation of functionality at 2.4 GHz : For the purposes of testing, we simulate the strongARM latch at a clock frequency of 1 KHz. In the further section, we compose the strongARM latch as a part of the broader analog front-end, and simulate at the expected frequecy requested by downstream processing, 2.4 GHz.

D. Analog Front-end and USB 2.0 Requirement Verification

The strongARM comparator served as the backbone for our design. However, the AFE not only consisted of the strongARM comparator, but also of two resistors R_{pd} conencted to ground. We've attached an image of the AFE schematic below:

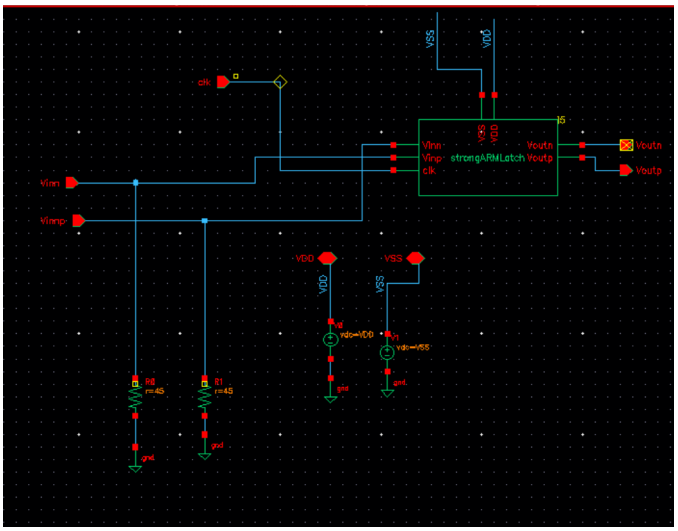

Fig. 7. Image of the analog front-end. The design contains the (1) strongARM latch and (2) pull down resistors with 45 ohms of resistance in order to remain spec-compliant.

We now turn our attention towards addressing the requirements listed out for our AFE in the USB 2.0 spec. We then perform additional testing with the target of verifying spec compliance.

1) **Minimum, Input Side Impedance**: As demonstrated in the complete AFE schematic in Figure 7, the analog-front end has two 45 Ohm resistors attached.
2) **Common Mode Voltage Resilience**: For this, we perform additional verification, adding a common voltage to the $D+$ and $D-$ signals in our testbench. Two common mode voltages - $500mV$ and $-5mV$ - defined the range of variation for the differential input signals. As a worst case, we alter our AFE test simulations, largely identical to the strongARM simulations above, to shift the differential input voltages by the two extremes, and ensure performance is as expected.
    a) The waveforms shown here in Figure 8 demonstrate the output strongARM successfully performing a comparison at the $500mV$ extreme. Testbench shown here -Figure 9
    b) A similar testbench was used for $-5mV$ common mode shift to confirm USB specification compliance, yielding similar waveforms.

Fig. 8. Waveform simulations for the analog frontend, at a frequency of 2.4 GHz. This verifies the strongARM performance at the target frequency, but also demonstrates its resilience to a phase shift of $500mV$, applied to both $D+$ and $D-$ signals.

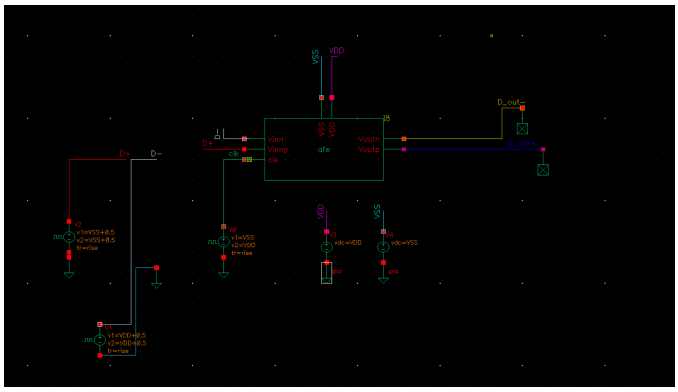

Fig. 9. Simulator for the $500mV$ common mode shift. Note the $+0.5V$ added to $D+$ and $D-$ pulse signals.

Overall, we proceed with confidence that our analog frontend complies with USB 2.0 specification requirements at the simulation level.

E. Strong ARM Latch Layout

As per timing constraints, we focused on converting the core part of our design - the strongARM latch - into a layout that could be exported to an integration team. This process would require multiple steps - (1) implementing a layout, (2) ensuring the layout passes the Sky130 PDK's design rule checks (DRC), (3) ensuring that the final layout matches the schematic we carefully designed above, and (4) exporting GDS, LEF, and LIB files to the chip integration team to add to the chip design.

F. Layout Design

The layout shown above in Figure 10 was the final physical design of the strongARM latch we ended up making to export to the integration team.

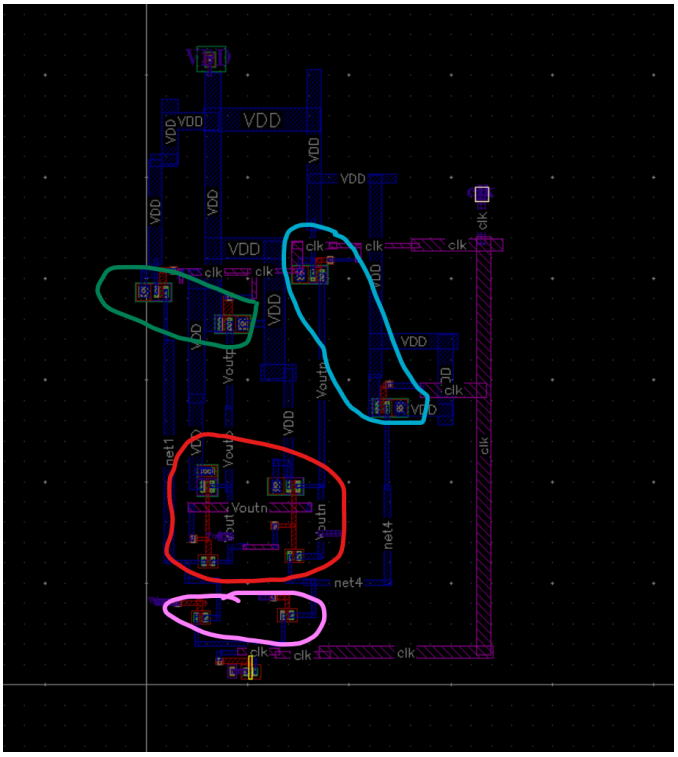

Fig. 10. Annotated examples of key areas of the strongARM latch. Red shows the cross coupled inverters, blue and green represent the left and right side precharge transistors, and the bottom shows the NMOS access transistors.

The 11-transistor layout in Figure 10 was designed by using $metal1$ as the primary routing and pin layer, and $metal2$ to route the clk signal to the various NMOS and PMOS transistors. Tap cells provided by standard packages were injected to provide power connection to the body of the transistors.

VI. Results

A. Performance and Area Statistics

Metric	Value	Spec	Third party	Spec met?
Area	525 μm^2	NA	17 μm^2	Y
Comparator Delay	273 ps (sim)	0.41 ns	200 ps	Y

TABLE I
FINAL PERFORMANCE AND AREA STATISTICS.

Table I shares the latest performance and area statistics on the strongARM comparator block. Our area, according to post-integration data, appears to be $525um^2$. This is in stark comparison to third party implementations [6], which have a size of $17um^2$ post optimization. We recommend considering this number with a grain of salt, as they leverage a different process node - UMC $180nms$. It does motivate a desire to reduce the layout area however, and optimizations can very easily be made.

A key optimization that needs to be made to reduce area for this relatively unoptimized layout is to remove excessive $metal1$ routing to the tapcell at the top, and aggressively route the clk pin, which currently takes additional area by extending out to the bottom and right. Additional room for optimization finally lies in bringing the precharge and access transistors closer to the cross coupled transistors.

Additionally, we see that our comparator features a propagation delay of $273ps$ at the working clock frequency of 2.4 GHz. Comparatively, we find that a strong ARM third party comparator has a delay on the order of $100 - 200ps$. Opportunities for minimizing propogation delay lie within appropriately sizing the transistor gates, which ended up largely preserving their default values.

In the broader scheme of the spec, we see that the propogation delay doesn't become a problem when it comes to required spec timing. A clock frequency of $2.4Ghz$ suggests a minimum clock period of $0.41ns$, leaving $0.41 - 0.27 = 0.14ns$ of slack for further combinational elements and flip flop setup / clock-to-q.

For further discussion on direct USB requirement verification at the schematic level, please revisit Section Analog Front-end and USB 2.0 Requirement Verification.

B. Passing Design Rule Checks (DRC)

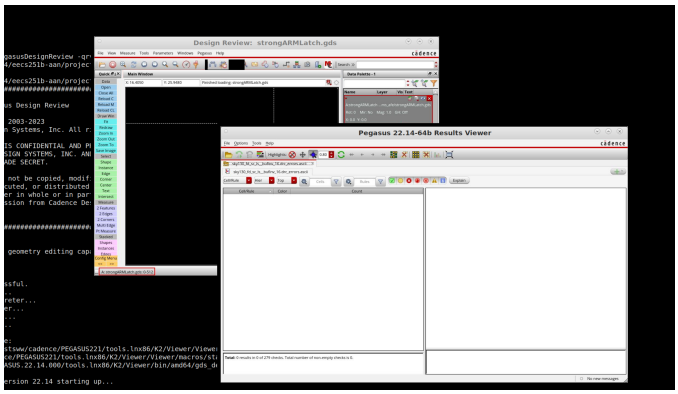

Fig. 11. Output from Pegasus DRC Results viewer that demonstrates the layout is DRC clean.

The figure above demonstrates compliance with the Sky130 V0.0.4 design rule deck. We faced some notable challenges in this process, and we've listed particular learnings for future reference

- **nwellTap must not be connected directly to the body**: A significant amount of time was devoted towards resolving this DRC constraint, which after future research turns out to be relatively standard. The Sky130 PDK provides options to extract a pfet's "tap" on the left/right/top/bottom. In order to successfully make a connection, a metal1 route must be placed from your power pin to the tapcell, without additional intermediaries such as licon to metal contacts.
- **licon to metal contacts for power cell routing**: All pfet and nfet terminals apart from tap cells must rely on a *licon to metal* routing contact.

- **polysilicon routing must cover the entire gate for it to pass DRCs**: This wasn't an obvious DRC to resolve, as the Sky130 DRC deck simply states that the "poly width must be greater than a certain value". This, combined with poly-to-poly distance DRC constraints, were one of the key challenges we faced during the layout validation process.

C. Passing Layout vs. Schematics Checks (LvS)

```
#####################################################################################################
#####
#####                                    Pegasus LVS COMPARISON
#####
#####  Version                    :   22.14-s007
#####  NVN Run Start              :   Tue May  6 23:30:24 2025
#####  ERC Summary File           :   result.sum
#####  Extraction Report File     :   strongARMLatch.lvsrpt
#####  Comparison Report File     :   strongARMLatch.lvsrpt.cls
#####  Top Cell                   :   strongARMLatch  <vs>  strongARMLatch
#####
#####  Run Result                 :   MATCH
#####
#####  Run Summary                :   [INFO]  ERC Results: Empty
#####                             :   [INFO]  Extraction Clean
#####
#####  Layout Design              :   /scratch/eda-4/eecs251b-aan/project/eecs251b_sp25_rx_ams_afe/stro
#####  Layout File                :   strongARMLatch.net (cdl)
#####  Schematic File             :   ../strongARMLatch.spice (cdl)
#####  Rules File                 :   /home/ff/eecs251b/sky130/sky130_cds/sky130_release_0.0.4/Sky130_L
#####  Pin Swap File              :   strongARMLatch.lvsrpt.cps
#####
#####  Extraction CPU Time        :   0h 0m 1s - (1s)
#####  Extraction Exec Time       :   0h 0m 6s - (6s)
#####  Extraction Peak Memory Usage :  133.00MB
#####  NVN CPU Time               :   0h 0m 0s - (0s)
#####  NVN Exec Time              :   0h 0m 1s - (1s)
```

Fig. 12. LVS results on the final exported layout. The layout is LVS clean, matching the functionality of the SPICE schematic extracted from Virtuoso simulation.

Figure 12 demonstrates compliance with the Sky130 V0.0.4 design rule deck. We've listed some of the key challenges faced during the process.

- **Uninterpretable, tap-related short circuits**: Unfortunately, limitations in Pegasus tooling prevents direct correlation between nodes in the schematic and nodes in the physical layout. These errors arose for all transistors, and it was resolved when the tapcell issue at the DRC stage was resolved.

D. Final Chip Integration Results

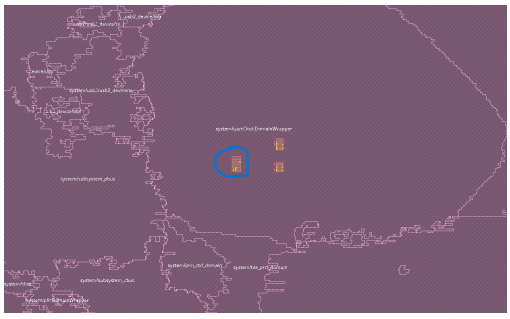

Fig. 13. The blue, circled instance showcases the integrated strongARM latch on the broader chip layout.

As shown in Figure 13, the strongARM latch was successfully integrated by an integration team into a chip implementing USB-2.

Conclusion

In this work, we successfully met the key USB 2.0 analog frontend requirements by implementing a differential receiver with 45 ohm termination resistances and demonstrating robust common-mode voltage tolerance across the -50mV to $+500\text{mV}$ range. Through Virtuoso simulations at target speeds up to 480 Mb/s (2.4 GHz sampling), our StrongARM comparator-based design exhibited correct decision behavior, sub-300 ps propagation delay, and full functionality under worst-case common-mode shifts.

Overall, the project encompassed end-to-end analog design tasks: selecting and sizing a zero-static-power StrongARM latch topology, integrating 45 pull-down networks for bus termination, and validating performance across low-speed, full-speed, and high-speed modes. We laid out the 11-transistor comparator in the SkyWater 130 nm PDK, resolved critical DRC challenges around tap-cell connections and poly routing, and verified LVS correspondence to our schematic. Post-layout statistics showed an area of 525um^2 and timing slack of 0.14ns against the 0.41ns USB 2.0 bit-period, affirming margin for downstream digital recovery circuitry.

This hands-on exercise provided deep insight into the trade-offs between speed, sensitivity, and input impedance in mixed-signal front-ends, and reinforced practical skills in floorplanning, parasitic management, and foundry-compliant layout verification. By delivering a DRC- and LVS-clean GDS, LEF, and LIB export, we enabled seamless integration of our analog block into a full USB-2 tapeout flow, rounding out a comprehensive learning experience in modern PDK-based analog/mixed-signal design.

Acknowledgment

I would like to thank Professor Borivoje Nikolic, Dr. Richard Dorrance, and Di Wang for their generous support throughout this project process.

Code and Reproducibility

Please find the original development GitHub repository at this link: https://github.com/AnshKetchum/eecs251b_sp25_rx_ams_afe. For the PR into the broader USB chip, see https://github.com/ucb-eecs251b/sp25-project-chipyard-usb2/pull/23

References

[1] J.-J. Nam, Y.-J. Kim, K.-H. Choi, and H.-J. Park, "A utmi-compatible physical-layer usb2.0 transceiver chip," in *IEEE International [Systems-on-Chip] SOC Conference, 2003. Proceedings.*, 2003, pp. 309–312.

[2] A. Alshehri, M. Al-Qadasi, A. S. Almansouri, T. Al-Attar, and H. Fariborzi, "Strongarm latch comparator performance enhancement by implementing clocked forward body biasing," in *2018 25th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, 2018, pp. 229–232.

[3] C. Chen, "Self-calibrating on-chip interconnects and noise-reduction techniques for clocked comparators," Ph.D. dissertation, Stanford University, 2010, available online at https://vlsiweb.stanford.edu/people/alum/pdf/1203_Chen_Self-Cal_Interconnects.pdf. [Online]. Available: https://vlsiweb.stanford.edu/people/alum/pdf/1203_Chen_Self-Cal_Interconnects.pdf

[4] K. M. Lei, P.-I. Mak, and R. Martins, "Systematic analysis and cancellation of kickback noise in a dynamic latched comparator," *Analog Integrated Circuits and Signal Processing*, vol. 77, pp. 277–284, 11 2013.

[5] Texas Instruments, “TUSB1210 Hi-Speed USB Device Transceiver with UTMI Interface,” https://www.ti.com/lit/ds/symlink/tusb1210.pdf, 2009, revised July 2021. [Online]. Available: https://www.ti.com/lit/ds/symlink/tusb1210.pdf

[6] S. Li, Z. Xu, and T. Iizuka, “Analysis of strong-arm comparator with auxiliary pair for offset calibration,” *Analog Integrated Circuits and Signal Processing*, vol. 110, no. 3, pp. 535–546, 2022. [Online]. Available: https://link.springer.com/article/10.1007/s10470-022-01992-6