Design of High-Speed 8-bit Leaky TReLU (Project 3)

Justin Troth
University of Minnesota CSE
EE5323: VLSI Design, Fall 2025
Instructor: Prof. Yu Cao
troth004@umn.edu

***Abstract*—This project presents the design and analysis of an 8-bit leaky TReLU module optimized for high-speed VLSI implementation. The module processes an 8-bit input x and a 4-bit threshold θ, producing a Q8.3 fixed-point output. A combination of subtraction, predictive addition, and multiplexing is used to implement the piecewise leaky TReLU function efficiently, minimizing transistor count and parasitic capacitance. Simulation on the TSMC 14 nm FinFET educational PDK in Cadence Virtuoso shows a minimum clock period of 53 ps with 460 transistors. Various full-adder topologies, carry-skip, and comparator-based alternatives were evaluated, demonstrating the trade-offs between speed, area, and critical path optimization.**

I. Introduction

Leaky TReLU (Thresholded Rectified Linear Unit) is employed to reduce switching activity and improve energy efficiency in digital neural network hardware. By allowing a small, nonzero slope for negative activations, it mitigates the "dying neuron" problem while suppressing low-magnitude transitions, thereby reducing unnecessary computation. This nonlinear behavior preserves inference capabilities while improving robustness. These properties make leaky TReLU particularly suitable for VLSI implementations where both power and throughput are critical.

In this project, the objective is to design a leaky TReLU unit optimized for maximum operating speed while minimizing area.

II. Methodology

Design was performed using the TSMC 14 nm FinFET educational PDK in Cadence Virtuoso. The supply voltage was set to $V_{\mathrm{DD}} = 0.8\,\mathrm{V}$, and the minimum fin sizing was 4 fins per transistor. Temperature was set to the default of 27°C. The NMOS and PMOS models used were `nch_lvt_mac` and `pch_lvt_mac`, respectively. Logical effort sizing was not applied according to course guidelines, and all transistors were sized to a minimum of 4 fins. All input signals, including the clock, had a rise and fall time of 20 ps. All output flip flops have a 16 fin inverter (shorted to ground) placed on their inputs to simulate a load. Spectre was used for simulation. HSpice was also used for verification.

The module accepts an 8-bit input x and a 4-bit input θ. The output f is a positive binary value in Q8.3 fixed-point format. For a leaky TReLU, these values follow the equation:

$$f(x) = \begin{cases} x & x > \theta \\ \alpha(x-\theta)+\theta & x \le \theta \end{cases} \tag{1}$$

Here, α is the threshold parameter and is set to 0.125 according to the project specification. This is a piecewise operation, which can be implemented using a multiplexer. In the proposed topology, the multiplexer control signal is derived from the output of a subtraction operation. By optimizing the addition logic, the number of required operations can be significantly reduced. The goal is to avoid an 8-bit adder, as it would introduce considerable delay.

The subtraction operation $(x-\theta)$ is unavoidable. We refer to this lower portion of the operation as y. The three least significant bits, lower$[2:0]$, are not involved in the addition with θ, so they can be ignored at this stage. The intended operation is:

+	θ_3	θ_2	θ_1	θ_0
	1	1	1(sign)	y_3

TABLE I: 4-bit addition example.

The table above represents another 4-bit operation. The sign of the lower bits is already known: this branch is only taken when $x < \theta$, and after computing $x-\theta$, the output is negative. Therefore, the lower sign bit must be set to 1. To add this negative number, sign extension is required.

As it stands, the improvement in speed is minimal, since two 4-bit adders are still chained together. However, by using y_3 as the control signal for an additional multiplexer, we can precompute the values of $\theta + 0b1110$ and $\theta + 0b1111$. This yields:

+	θ_3	θ_2	θ_1	θ_0
	1	1	1	0

+	θ_3	θ_2	θ_1	θ_0
	1	1	1	1

TABLE II: Branching additions for the upper part.

Because there are several known values in these additions, we can save on circuitry, and more importantly speed due to reducing capacitance. z_n refer to the upper four bits of the output. The corresponding outputs are listed below:

These operations can be easily implemented in hardware with less delay and parasitic capacitance than two full adders. The carry bit is not a concern, since the maximum output ($x =$

$y_3 = 0$	Output
z_0	θ_0
z_1	$\neg\theta_1$
z_2	$\theta_2 \oplus \neg\theta_1$
z_3	$\theta_3 \oplus NOR(\theta_1, \theta_2)$

$y_3 = 1$	Output
z_0	$\neg\theta_0$
z_1	$\theta_1 \oplus \neg\theta_0$
z_2	$\theta_2 \oplus NOR(\theta_0, \theta_1)$
z_3	$\theta_2 \oplus NOR(\theta_0, \theta_1, \theta_2)$

TABLE III: Logical operations equivalent to 4-bit adders.

$15, \theta = 15)$ is 15, which in Q8.3 format is $0b0000\,1111\,000$, occupying only the first seven bits. The two "adders" can also share some gates. Because these "adders" compute faster than the subtraction, the number of transistors can be reduced without affecting the critical path. The reduction in transistors also lowers node capacitance, improving the speed of the critical path.

When $x > \theta$, x is shifted left by 3 to fit the Q8.3 format.

Figure 1 shows an elaborated design generated in Vivado using Verilog to execute the same process described above. The predictive adders have not been fully optimized but still resemble the final implementation (excluding registers), as seen in Figure 2. The two multiplexer stages are prominent, along with the subtraction module.

Gates with fewer than two inputs are implemented using transmission gate logic, while gates with three or more inputs use static logic. The 4-bit multiplexers consist of four 1-bit multiplexers sharing the same control signal and are implemented with transmission gates. The subtraction module is implemented as a ripple carry adder, with Cin held high and one input inverted. Full adder modules use 16-T transmission gate implementations [1].

Flip-flops are placed on both the input and output. Eleven-transistor True Single-Phase Clock (TSPC) flip-flops are used to maximize speed and minimize area. Transmission AND gates replace some multiplexers in the end-select portion of the circuit when inputs remain constant. For example, when $x > \theta$, the three least significant bits are always 0. Thus, these outputs only switch to 1 when the select signal is low and the subtraction result is high, saving transistors.

The circuit was verified by checking all 256 combinations of $\theta \in [0, 15], x \in [0, 15]$. When x is larger than 15 the result is always $x << 3$ and the critical path becomes much shorter. A python script was used to check the voltage of the output one clock cycle after input change and check against a leaky TReLU implemented in code. Transitions were tested with all initial voltages at VSS and at VDD, for a total of 512 test cases.

III. Results

The fastest clock period was measured to be 53 ps. When running at 56 ps clockm, this consisted of approximately 10 ps for clock-to-Q, 41 ps for combinational logic, and 5 ps for setup time. We can estimate that combinational logic takes at most 38ps, since the circuit still passes tests at a 53 ps clock period. The circuit was implemented with a total of 482 transistors, or 460 transistors when excluding the inverters acting as loads for each of the output flip flops.

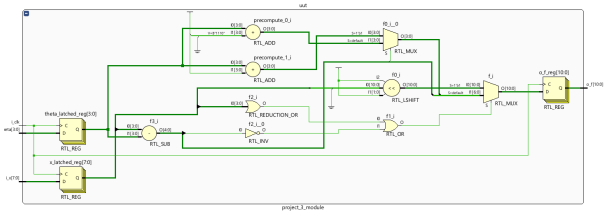


Fig. 1: Elaborated design generated in Vivado

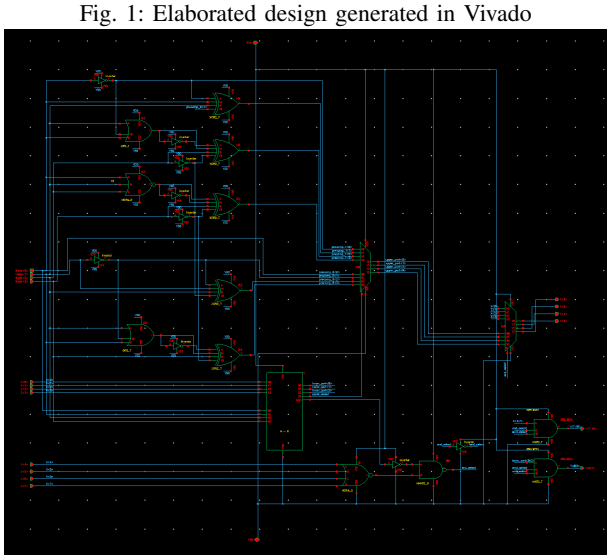


Fig. 2: The proposed combinational circuit in Cadence Virtuoso

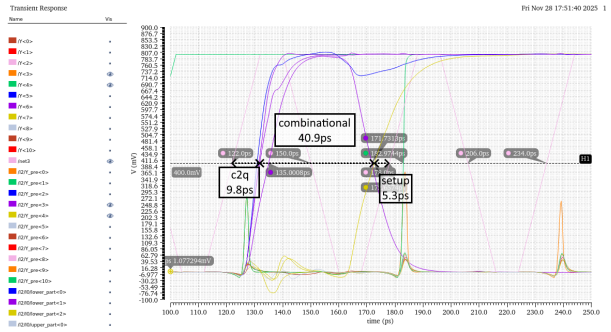
The slowest transitions were observed at Y_3 and Y_4, which originate from the upper half of the circuit. As shown in Figures 3a and 3b, the output can rise and then fall, due to the multiplexer controlling the predictive upper bit (driven by y_3) switching when the predicted values differ. This VDD-to-VSS transition takes time and propagates through the end-select multiplexer, forming part of the critical path.

IV. Discussion

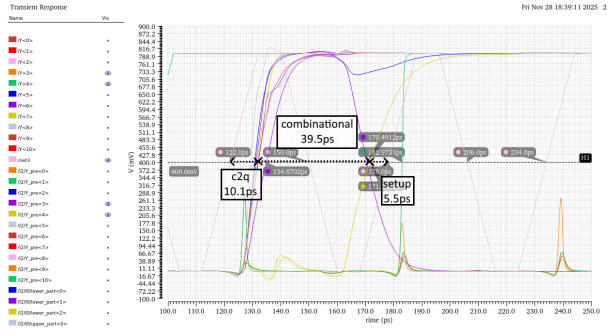
Several full adder topologies were evaluated. The traditional 26-T full adder was found to be 15 ps slower than the 16-T transmission gate full adder when implemented in the complete combinational circuit. A hybrid full adder proposed by Bhattacharyya et al. [2] was also tested, but it was 26 ps slower than the baseline.

A 4-bit carry skip adder was tested; however, the path through the skip section was longer than the ripple through four adders. Carry look-ahead logic was also examined, but despite its theoretically shorter critical path, the increased gate count led to higher parasitic capacitance, resulting in slower performance than the ripple carry adder.

A comparator can have a shorter critical path than a subtraction module, but when directly replacing the "Sign" bit from the 4-bit subtraction module, it was slower. Additionally, y_3 from the subtraction module takes nearly as long to compute and forms another part of the critical path. Replacing "Sign" with a comparator would require accelerating y_3 as well, while



(a) Timing diagram for test case $\theta = 3$, $x = 1$



(b) Timing diagram for test case $\theta = 15$, $x = 13$

Fig. 3: Timing diagrams for the two test cases at clock = 56 ps

carefully managing transistor count to minimize parasitic capacitance.

Using pass-gate logic instead of transmission gates could potentially reduce transistor count, but may be slower due to reduced noise margins. Preliminary tests showed instability, making it difficult to conclude whether pass gates would outperform transmission gates without further study.

Buffers were added before the output flip flops, as the rules allow resizing of these buffers for to minimize logical delay. Adding buffers, despite proper sizing, was found to increase the delay, and as such the buffers were removed.

Potential improvements include converting large static gates to dynamic gates, which could significantly reduce parasitic capacitance and overall delay. Sizing based on logical effort could also provide meaningful gains, although it was not allowed in this project.

Swapping logically equivalent inputs can improve timing. For example, two subtraction circuits were tested: A - B and B - A. B - A was an entire picosecond faster when tested because it removed an inverter from the critical path. Additionally, all transmission-style gates have this effect because one input is passed through while the other is used as a control signal. In testing, differences of around 0.3 ps were observed when changing the input signals for a XOR2 transmission gate. With more time and testing, these minor gains may provide a significant decrease in delay.

It is important to note that this circuit is not realistic for practical implementation. Layout has not been performed, so the impact of closely spaced, rapidly switching nodes cannot be evaluated. Cross talk must be considered in such a dense circuit. The design also assumes a perfect clock with no deviation and exactly 20 ps rise and fall times. Corner cases and temperature variations were not tested, which would affect real-world performance.

Power consumption is another concern. The design exclusively uses low-threshold voltage transistors, which increases power usage. Additionally, switching at a 53 ps period corresponds to an 17.8 GHz clock, which is impractical except under extreme conditions. For a fully implementable design, timing should be targeted for achievable frequencies on modern chips (6 GHz or lower) while optimizing for power and area.

V. Conclusion

An 8-bit leaky TReLU unit was implemented using optimized combinational logic to reduce transistor count and critical path delay. By precomputing upper-bit additions and leveraging multiplexers controlled by the subtraction result, the design minimizes parasitic capacitance while maintaining correct functionality. The final circuit achieved a simulated fastest clock period of 53 ps with 460 transistors, demonstrating the effectiveness of predictive adder logic and transmission-gate full adders in high-speed VLSI design. Future work should focus on realistic layout implementation, dynamic gate optimization, and power-efficient design to ensure practical operation at achievable clock frequencies while maintaining low area and energy consumption.

References

[1] S. Shewale and S. Shirsath, "Design and analysis of cmos full adder," *International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)*, vol. 9, no. 1, 2021.

[2] P. Bhattacharyya, B. Kundu, S. Ghosh, V. Kumar, and A. Dandapat, "Performance analysis of a low-power high-speed hybrid 1-bit full adder circuit," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 23, no. 10, pp. 2001–2008, 2015.