8-bit Kogge-Stone Adder Design Report

Nick Ji (A18060321), Eric Truong (A18104696), Cleavant Yang (A17464613)

In this project, we set out to design a high-speed 8-bit adder using the Kogge-Stone architecture. Our main challenge was to implement a parallel-prefix adder that minimizes delay and preserves signal integrity at the transistor level, using static CMOS design in Cadence. High-performance adders are critical in digital systems such as ALUs, DSPs, and processors, where addition is often on the critical path—so improving its speed directly benefits overall system performance.

We initially considered traditional adder architectures, including ripple-carry, carry-lookahead, and carry-skip adders. However, these are limited by sequential or partially parallel carry propagation. Ripple-carry suffers from linear delay with bit width, and while carry-ahead and carry-skip improve this, they still present bottlenecks in high-order bit propagation.

To address this, we selected the Kogge-Stone Adder (KSA), which is one of the fastest known parallel-prefix architectures. KSA computes all generate(s) (G) and propagate(s) (P) signals in parallel at the start and then forms a tree structure to compute carries efficiently. This architecture offers a logarithmic delay and is highly parallelizable, making it optimal for our speed-focused design goals.

Our implementation incorporated several innovative elements to further enhance performance and reduce transistor count. We used transmission gates to implement XOR logic for both propagate and sum calculations, minimizing delay and transistor overhead. For Group Generate logic, we implemented an AOI21-based design to improve speed. Grey cells were used selectively in the prefix tree to reduce transistor usage without compromising performance. We also used piecewise linear (PWL) sources in our testbenches to better emulate real-world irregular signal conditions in Cadence. To further optimize the circuit, we modified the standard black cell by creating two variants, one with inverting outputs and one with non-inverting outputs. By strategically placing these along the prefix tree, we were able to eliminate unnecessary inverters and reduce the overall transistor count significantly during signal propagation between black cells. This architectural-level optimization allowed us to maintain correctness while improving area, power efficiency, and delays.

Figure 1 shows the standard black cell, implementing both the group generate and propagate functions. To optimize interconnect efficiency, we created two variants: the inverting-output black cell (**Figure 2a**) and the non-inverting-output black cell (**Figure 2b**). These variants allowed us to alternate complementary signal levels across the prefix tree, significantly reducing inverter usage. **Figure 3** presents the propagate-generate logic cell, which produces the initial P and G values from inputs A and B. The complete 8-bit Kogge-Stone Adder schematic (**Figure 4**) demonstrates how these cells were assembled into a full prefix network with sum logic. Our design was tested in Cadence using inputs that would propagate through the critical path (**Figure 5a/b**), with various iterations alongside their improved delays showcased in **Figure 6**. Finally, **Figure 7** summarizes key performance metrics and comparison to a regular ripple-carry adder, including maximum clocking frequency, energy consumptions, etc.

For future iterations, we suggest exploring dynamic logic styles or integrating DFFs to help align timing and suppress glitches. We also believe further optimization of the sizing and layout could yield even better results.

Nick designed the full 8-bit KSA, including the black cell, grey cell, PG logic, buffers, sum logic, and final assembly, and performed all transistor sizing and primary simulations.

Eric hand-drew the transistor-level schematic and designed the two black cell variants to reduce transistor count across the prefix tree.

Cleavant validated circuit functionality using Cadence and other simulation tools and developed dynamic testbenches to test signal behavior under varying conditions.

Acknowledgments:
Very special thanks to Professor Mercier for the opportunity.

References:

[1] Ramasamy, Senthil G. "Design and Analysis of Kogge-Stone and Han-Carlson Adders in 130nm CMOS Technology", Mar. 2018, *International Journal of Research*, 05(07):1063-1068

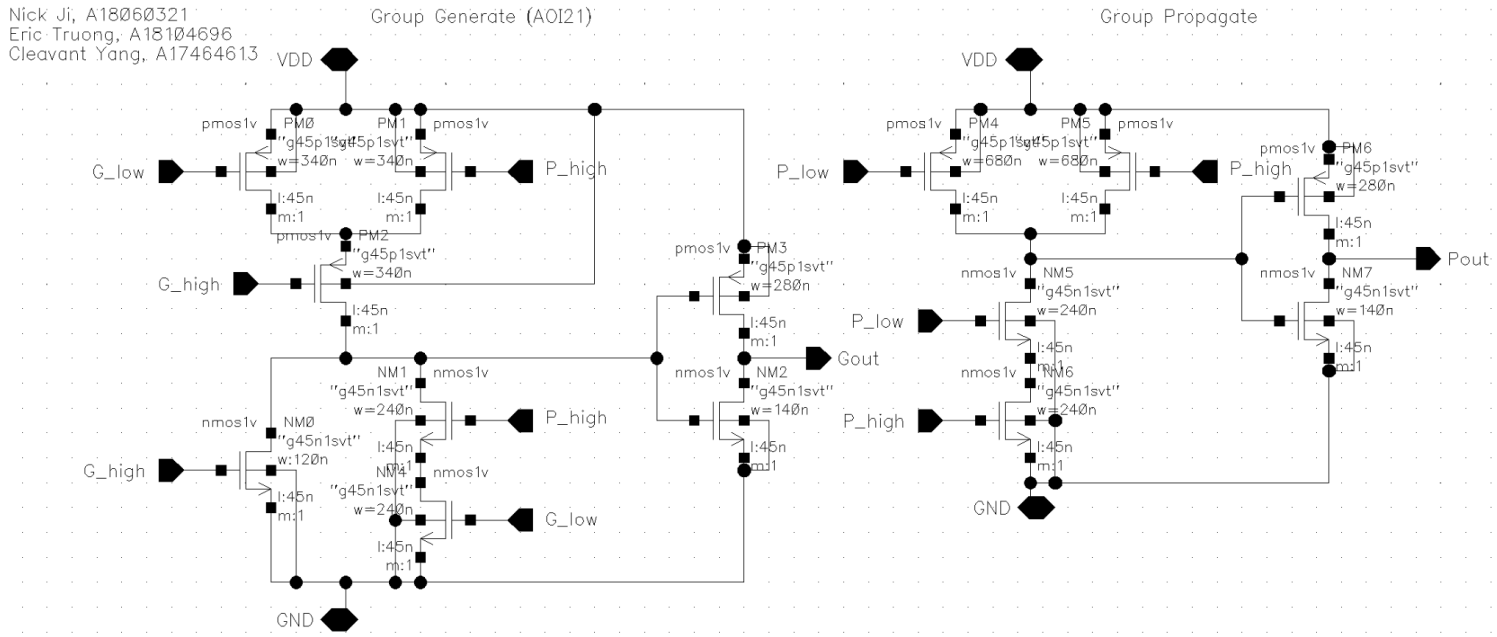

Figure 1. Standard Black Cell

The standard black cell used in Kogge-Stone adder designs implements the group generate function $G_{i:j} = G_i + P_i G_j$ (left circuit) and group propagate $P_{i:j} = P_i P_j$ (right circuit) using static CMOS logic. These combine into a single symbol for hierarchical design. The grey cell is simply separated from black cell, containing only Group Generate functionalities. For our original black cell design, we solved for both the PUN and PDN while keeping noninverted inputs, then placed an inverter on the output to create a universal cell. To optimize transistor usage and signal flow across the prefix tree, we further developed two custom black cell variants and utilized them in the final design. In the next page, Figure 2a. shows a version with non-inverted inputs and inverted outputs (essentially ridding of the inverter at the output), and Figure 2b. shows a version with inverted inputs and non-inverted outputs. By alternating these variants at even and odd levels of the prefix tree, we minimized the need for extra inverters between stages. This allowed us to significantly reduce transistor counts without compromising signal correctness.

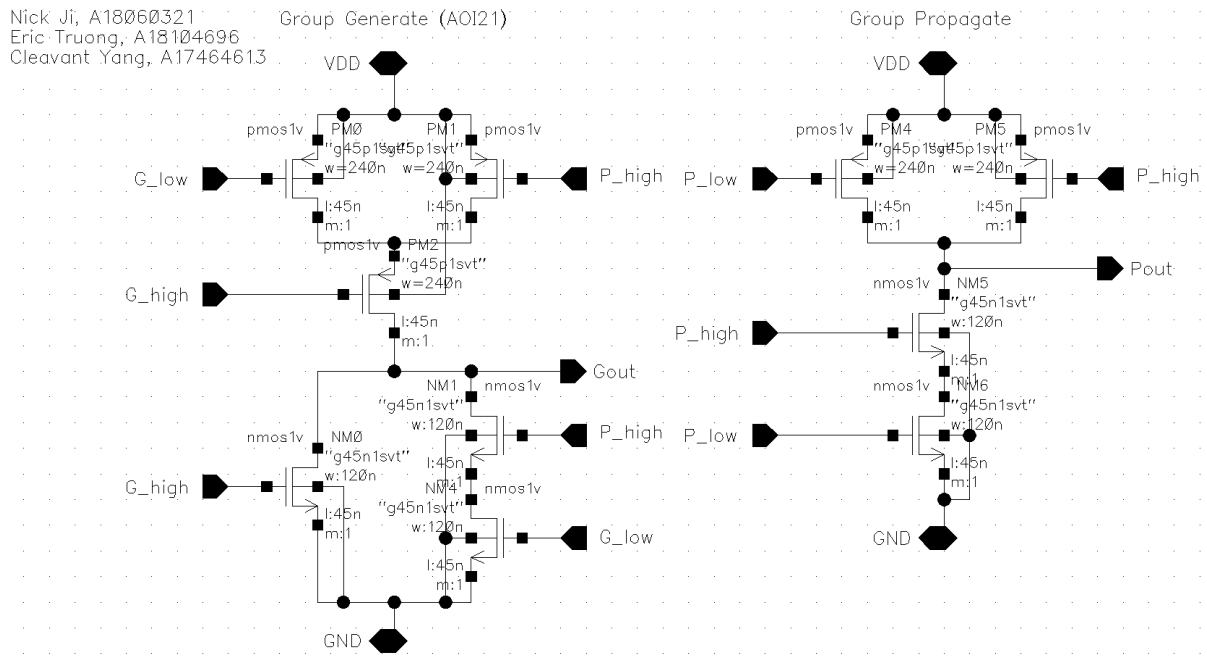

Figure 2a. Black Cell with non-inverted inputs

This variant shown the above figure receives non-inverted P_i, G_i and P_j, G_j inputs and produces inverted outputs. It is used in even levels of the prefix tree to reduce the need for external inverters.

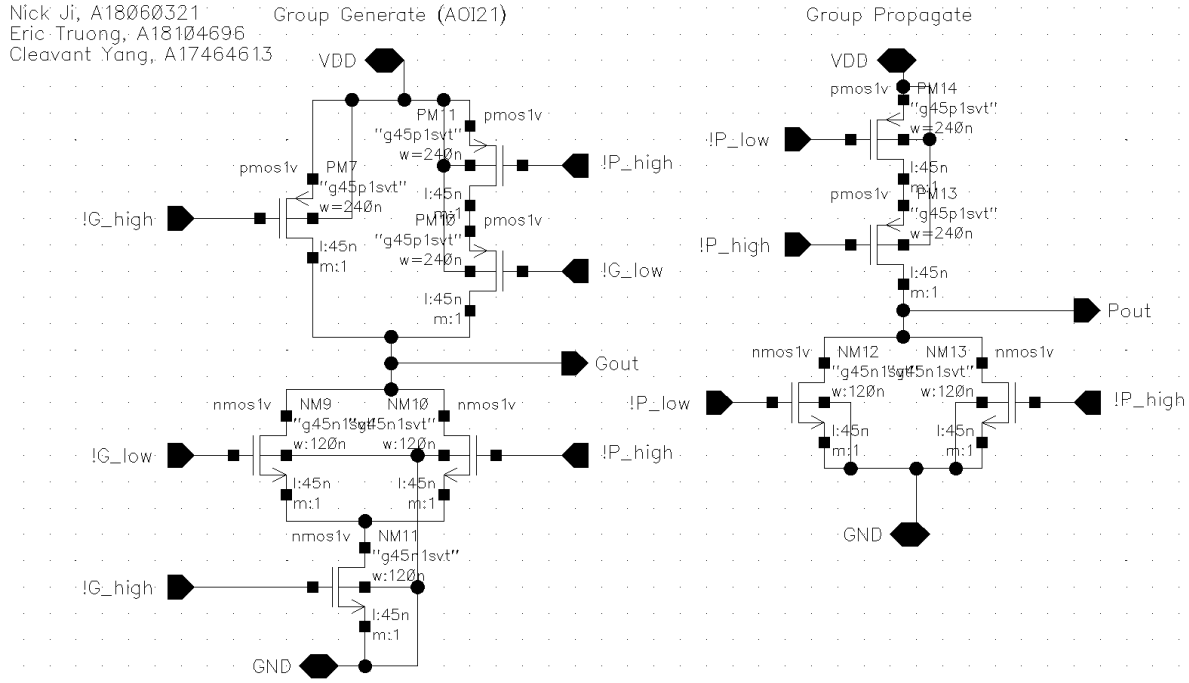

Figure 2b. Black Cell with inverted inputs

This variant receives inverted P_i, G_i and P_j, G_j inputs and produces non-inverted outputs. It is used in odd levels to restore non-inverted signals and complement the previous variant, maintaining logic correctness with fewer transistors.

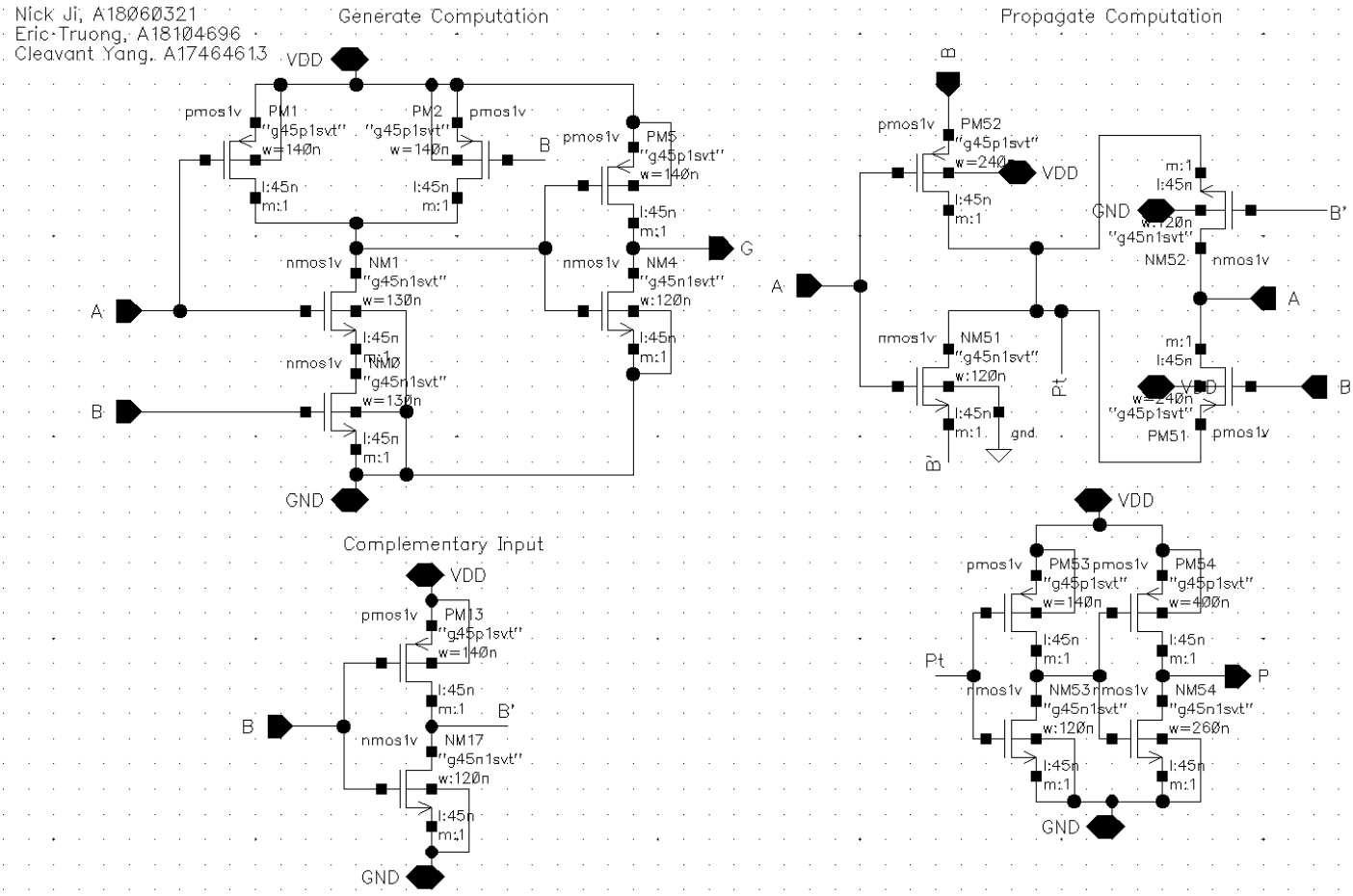

Figure 3. PG Cell

These are the PG blocks that compute the necessary propagates and generates to compute the overall sum. The sum cell implements the function $S = P \oplus C_{in}$ which is the same structure as propagate computation, using transmission gates and a buffer.

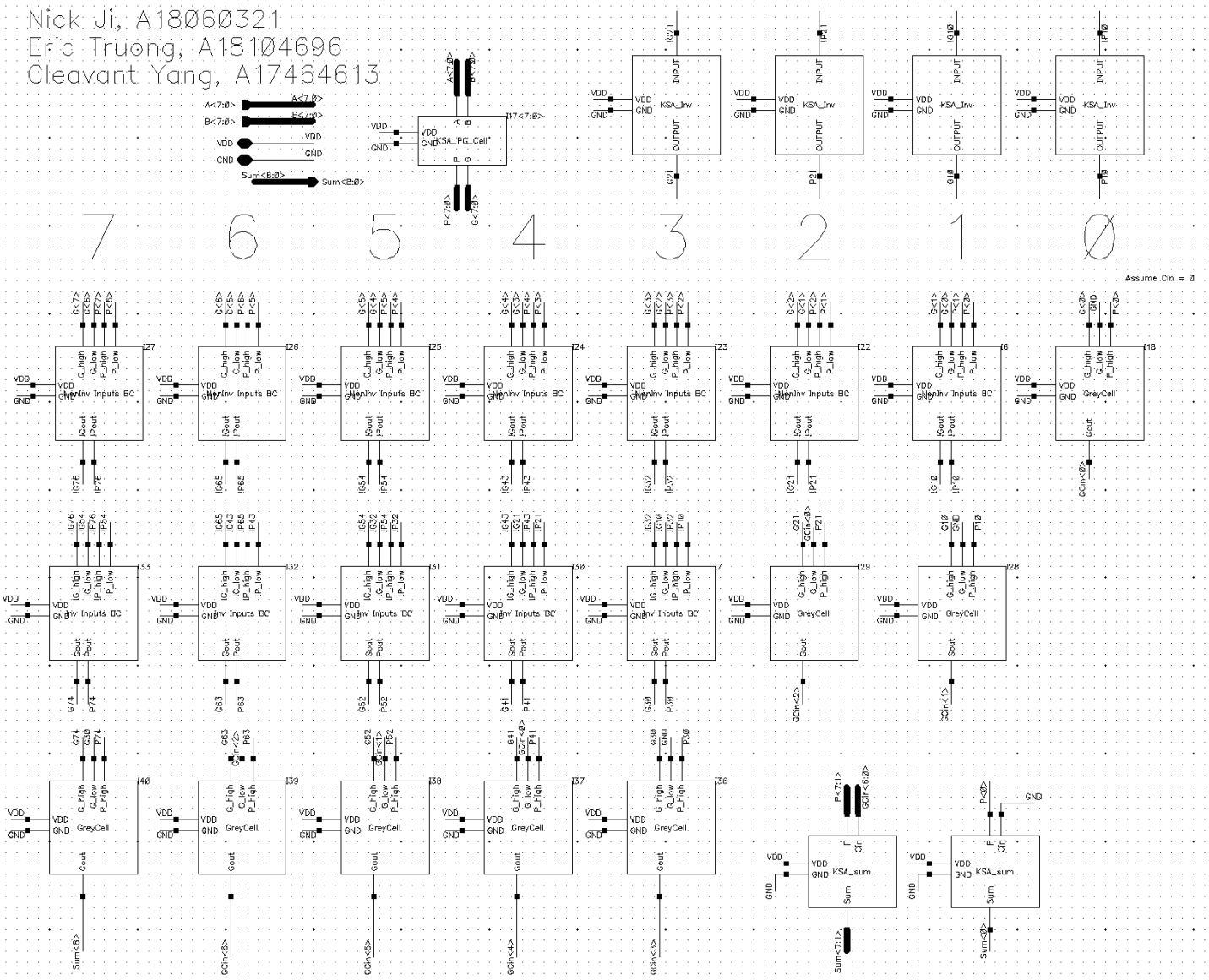

Figure 4. Complete 8-bit Kogge-Stone Adder

This figure shows the full schematic of the 8-bit Kogge-Stone Adder designed using static CMOS logic. The parallel-prefix structure is constructed using black and grey cells to compute carry signals in logarithmic time. Our design incorporates optimized black cell variants (inverting and non-inverting) to reduce transistor count by minimizing inverter use across levels. Buffers are added before the final sum logic (embedded within the "KSA_sum" block) to ensure strong output signal integrity. Sum outputs are computed using pre-generated propagate signals and the final carry-in values.

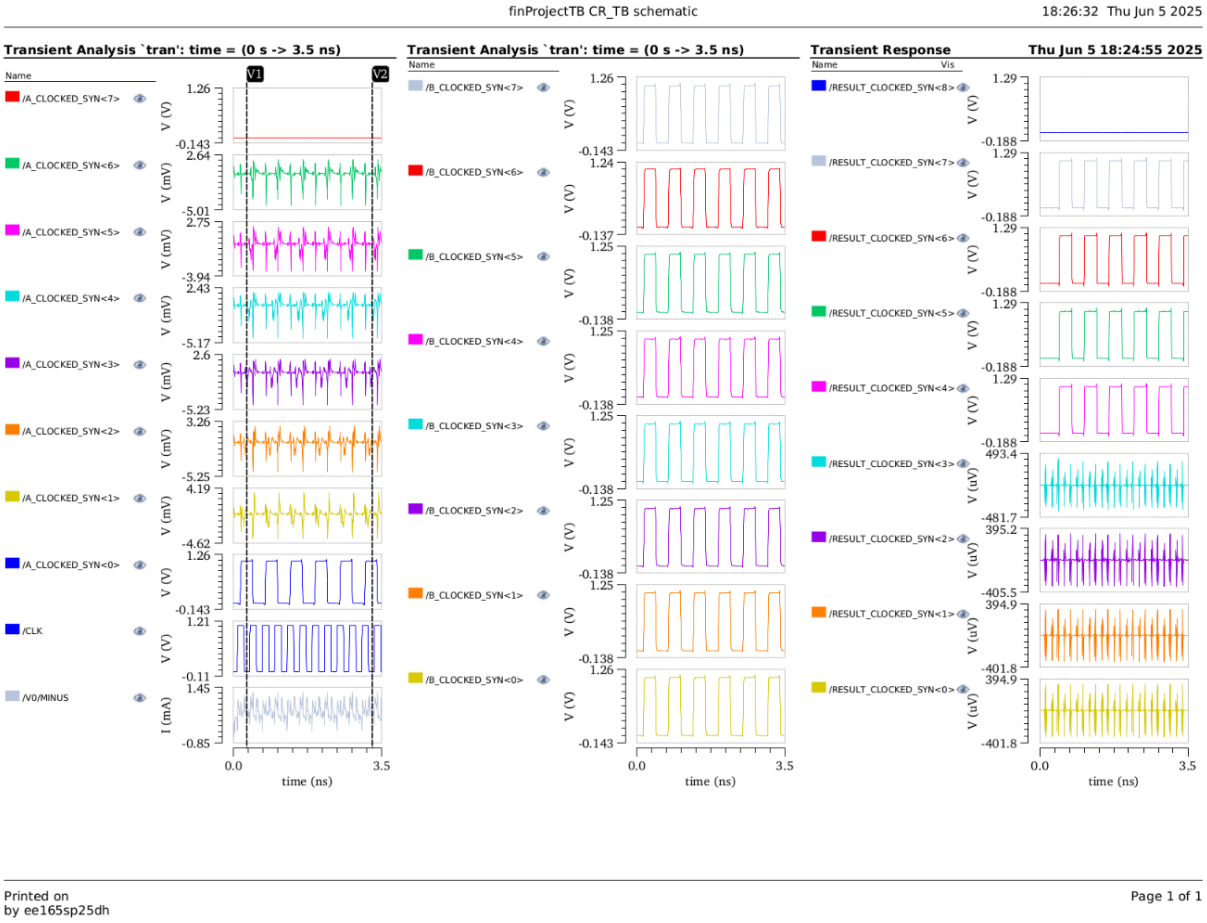

Figure 5a. Test Bench Critical Input Pair of Carry Ripple Adder at 3.4GHz

The above image shows the transient analysis of the clocked inputs and outputs of the carry ripple adder given the critical input pairs at the max frequency that the 8-bit KSA can function at, being 3.4GHz. As shown above in the first and second columns of graphs, A<0> is set to logic 1 and A<7:1> is set to logic 0 while B<7:0> is set to logic 1, representing the critical input pairs. The third column of graphs showcases the incorrectly added output bits. Power was measured through taking the clipped average over the ten clock cycles in between V1 and V2. A graph of the clock and total current measurement can be seen at the bottom of the first column of graphs.

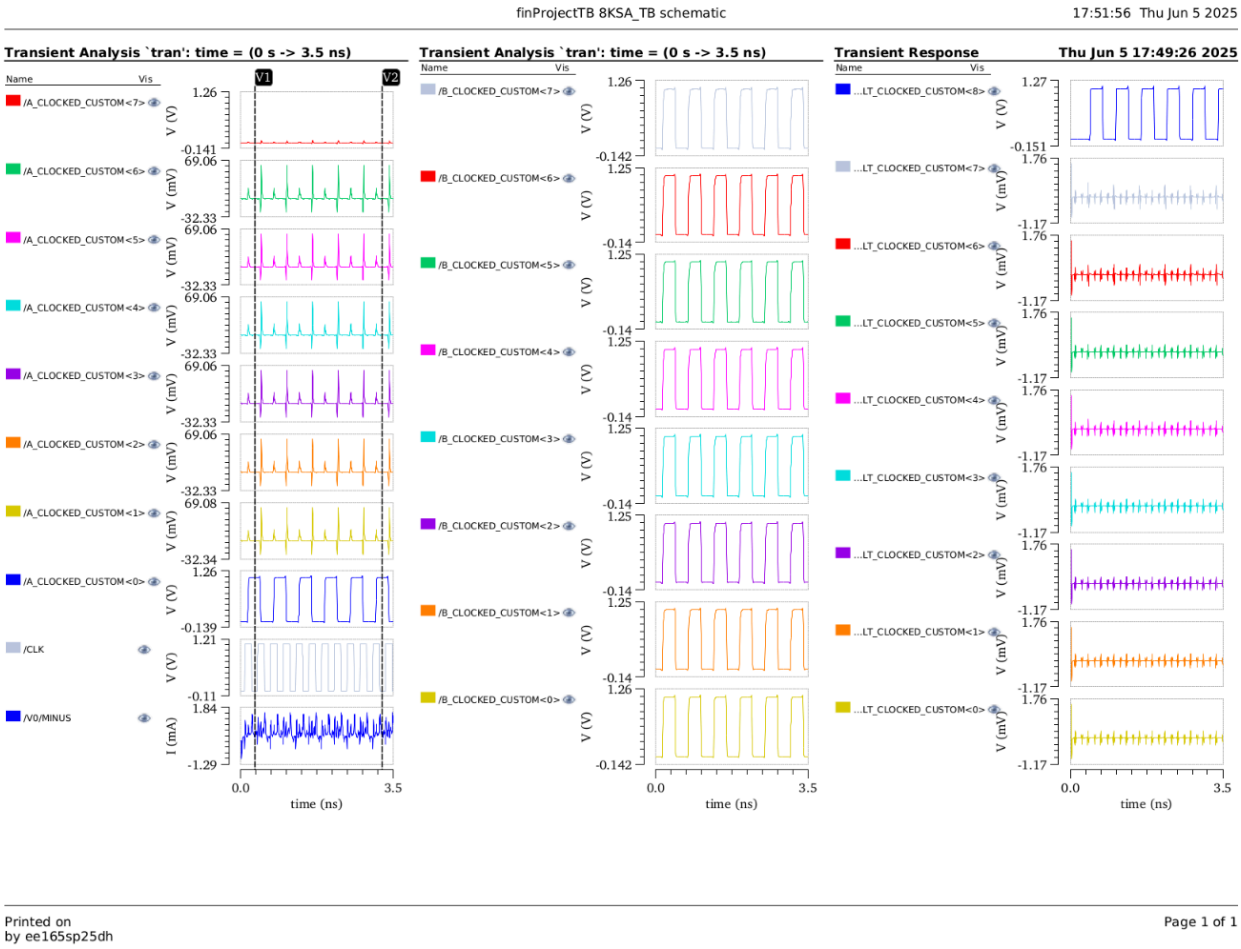

Figure 5b. Test Bench Critical Input Pair of 8-bit KSA at 3.4GHz

The above image shows the transient analysis of the clocked inputs and outputs of the 8-bit KSA given the critical input pairs at the max frequency it can function correctly at, being 3.4GHz. As shown above in the first and second columns of graphs, A<0> is set to logic 1 and A<7:1> is set to logic 0 while B<7:0> is set to logic 1, representing the critical input pairs. The third column of graphs showcases the correctly added output bits along with the clock to prove the frequency the adder is functioning at. Power was measured through taking the clipped average over the ten clock cycles in between V1 and V2. A graph of the clock and total current measurement can be seen at the bottom of the first column of graphs.

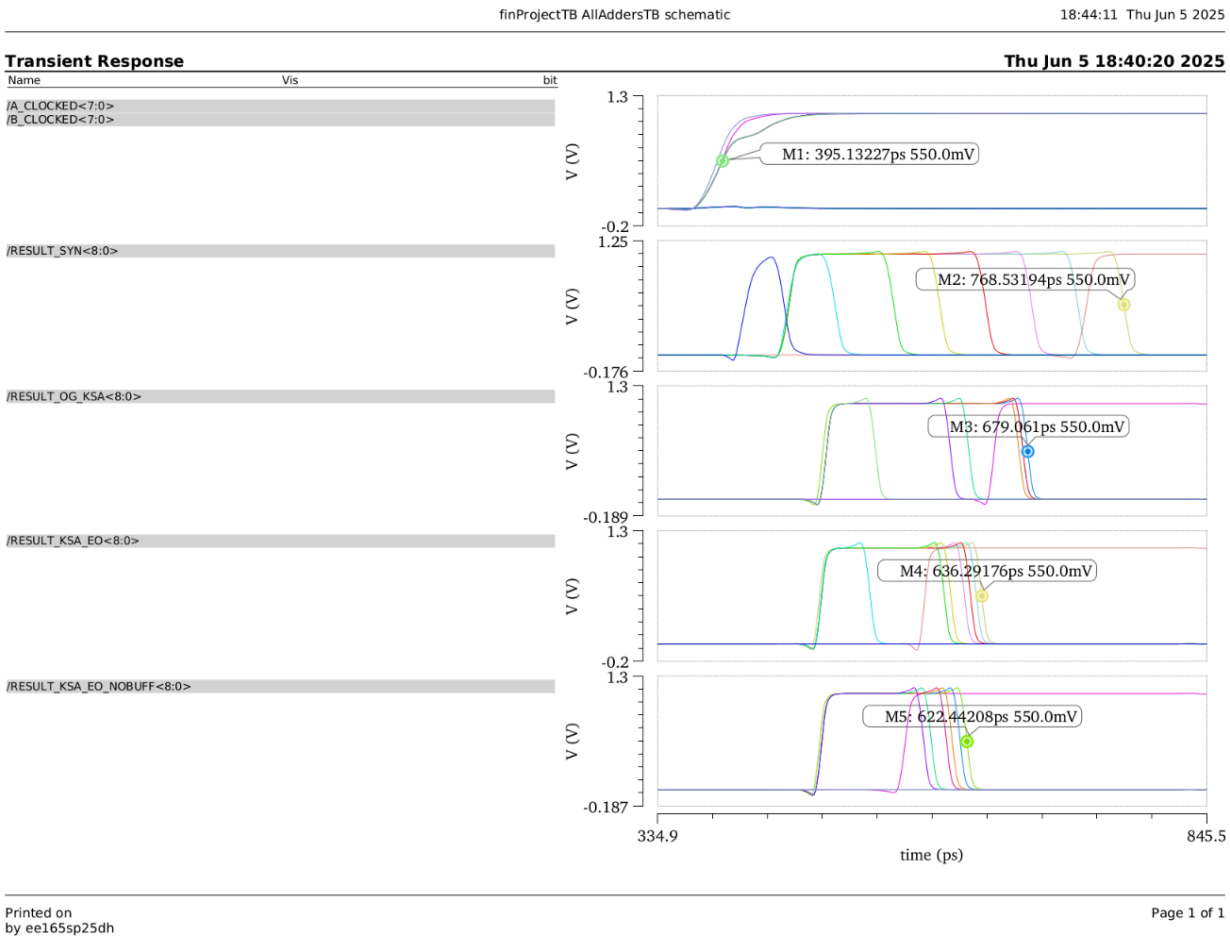

Figure 6. Propagation Delay

The figure above presents the propagation delay measurements for the Lab 4 adder alongside various iterations of our Kogge-Stone Adder (KSA) design. The second subgraph represents the delay of the Lab 4 adder, while the remaining subgraphs correspond to different versions of the KSA. The first KSA implementation "RESULT_OG_KSA" utilizes standard black cells without the incorporation of even and odd cell optimization in addition to buffers in between the sum computation block and the grey cells. The second version "RESULT_KSA_EO" introduces even and odd black cells throughout the design. In the final iteration "RESULT_KSA_EO_NOBUFF", buffer and standard black cells are removed and replaced with even and odd cells throughout, with an inverter added at the output to restore the correct polarity. As illustrated, each successive modification leads to a reduction in overall propagation delay, demonstrating improved performance.

Summary Table

	Place+route schematic	Custom design schematic	Place+route (optional)	Custom design extracted (optional)
	Performance for $V_{DD} = 1.1\,V$			
f_{max}	2.2 GHz	3.4 GHz	-	-
Power consumption @ f_{max}	323 uW	531 uW	-	-
Energy per operation @ f_{max}	146fJ	156fJ	-	-
	Performance for $V_{DD} = 1.1\,V, f_{CLK} = 1\,GHz$			
Power consumption @ 1 GHz	141 uW	145 uW	-	-
Energy per operation @ 1 GHz	141fJ	145fJ	-	-
	Other important parameters			
Adder architecture	Ripple-carry	Kogge-Stone	Ripple-carry	Kogge-Stone
Core area	-	-	-	-
Critical input pair (i.e., A = ?, B = ?)	$A = 00000001$ $B = 11111111$	$A = 00000001$ $B = 11111111$	-	-
Transistor types used (e.g., VTL, VTG)	RVT	RVT	-	-

Figure 7. Summary Table