8-bit Hierarchical Carry Lookahead Adder

Masaya Honda A15747299 and Damian Liang A12094494

The design challenge of this project is to create an 8-bit adder that is able to achieve a higher performance compared to the ripple carry adder that was previously implemented in Lab #4. As adders are used in a multitude of applications such as microprocessors, graphics related calculations, and even digital signal processing units, faster-performing adders are necessary for applications that demand performance.

There are many previously implemented architectures that are not optimal for speed when dealing with 8-bits such as the ripple carry adder and the standard carry-lookahead adder. The architecture of the ripple carry adder consists of cascaded full adders, where the last full adder that is responsible for the most significant bit is required to wait until the remaining full adders finish their computations. This inevitably results in a greater delay, as the carry bits need to be propagated from the first full adder to the n^{th} full adder. Thus, this architecture suffers a significant performance hit when dealing with more bits. The standard carry-lookahead adder performs faster than the ripple carry adder due to its architectural differences that include an n number of partial full adders that produce generate, propagate, and sum signals. This architecture falls short in applications that require a greater number of bits, such as 8-bits due to how the number of computed bits are directly proportional to the number of fan-ins within the lookahead logic.

Our chosen architecture is based on the carry-lookahead adder, but is implemented through a hierarchical process as shown in Figure 3. Specifically, we decided to implement the 8-bit carry-lookahead adder by using 4-bit carry-lookahead generator blocks that are responsible for computing the carryout, group propagate, and group generate bits. This design was inspired by the 16-bit carry-lookahead adder published by Simon Fraser University.[1] The hierarchical nature of this architecture will reduce the fan-ins in the look-ahead logic and allow for the delay to be related to the number of levels in the hierarchy.[1] Since the delay will no longer be related to the number of bits that it is calculating and how the maximum number of fan-ins in this architecture falls within the recommended range that was in our lecture, we can expect an improvement in the performance. This architecture also includes the sum block, which computes the sum, propagate, and generate bits based on the inputs A, B, and the carry-in. For the sum block, we utilized VTG transistors due to its balanced characteristics compared to the VTL and VTH counterparts. The schematic of this sum block is shown below in Figure 1, and the subsequent layout and its DRC and LVS reports are shown in Figures 2, 2a, and 2b. These sum blocks are used to output RESULTS<7:0>, while the top 4-bit carry-lookahead generator block is used to output the most significant bit, RESULTS<8>. This architecture allows for each of the sum blocks to directly compute the propagate and generate bits, which yields greater performance. As shown in Figure 5, comparing our custom design to the adder in Lab #4 after extraction, we were able to see that our design was able to reach a greater maximum frequency at the cost of power consumption. We observed and clipped the transient current waveform of each design when calculating the average power consumption, which is shown in Figures 4a, 4b, 4c, 4d to ensure accuracy. Based on the table shown in Figure 5, we can also see that the post-extracted results yielded a greater value for the energy per operation and lower maximum frequencies. This is due to the introduced parasitic capacitances that exist at the transistor level during the layout.

We ran into the issue of the final carryout always outputting a logic high regardless of the 8-bit inputs A and B. This 4-bit carry lookahead generator block is designed to take inputs from 4 propagate and 4 generate signals in order to output the necessary group propagate and generate bits that are used to determine the final carry out. However, in this final lookahead generator block, we only had 2 propagate and 2 generate signals coming in from 2 other generator blocks located just one level below in the hierarchy. So we needed to determine how to generate the last group propagate and generate signals based on these conditions. After analyzing the boolean logic functions, we determined that we needed to set permanent inputs to these unused input pins.

The most innovative aspects of our work are related to how we reduced the max fan-in in the logic gates from 4 down to 2 through boolean algebra. In addition, we used an XOR design that only uses 4 NAND gates, which significantly reduced the number of gates used and the subsequent layout area as shown in Figure 2. One iteration that could be made to improve this design would be to create a simplified lookahead generator block that is placed in the highest point in the hierarchy since it only needs to take in a total of 4 inputs instead of its designed 8 inputs that are used in the lower levels of the hierarchy. Another iteration could revolve around the idea of using an OR gate instead of an XOR gate that is used to compute the sum. The OR gate will consist of fewer gates, which in return will decrease the area of the design and potentially improve the speed. Throughout this project, we worked together during the key phases of brainstorming and implementation. We were both present when we checked and ensured that each of the schematics and their corresponding layouts functioned properly. Specifically, we both worked on all aspects of the project together either through taking turns doing the schematic drawing and performing the manual layout.

Acknowledgments: Nikhil Maganti and Rohit Madan

References:

[1] https://www2.cs.sfu.ca/CourseCentral/150/eyal/lectures/CLA.pdf

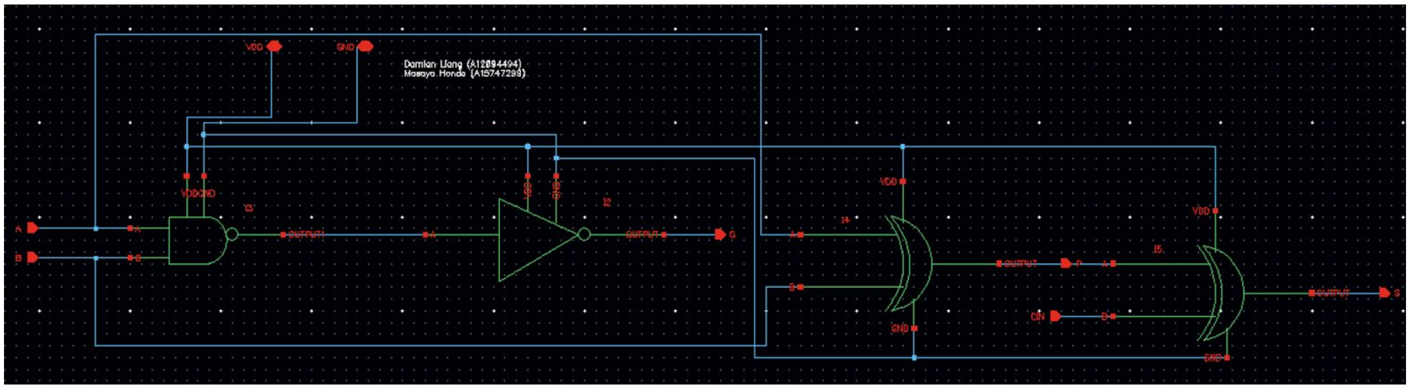


Figure 1 | Schematic of Sum Block

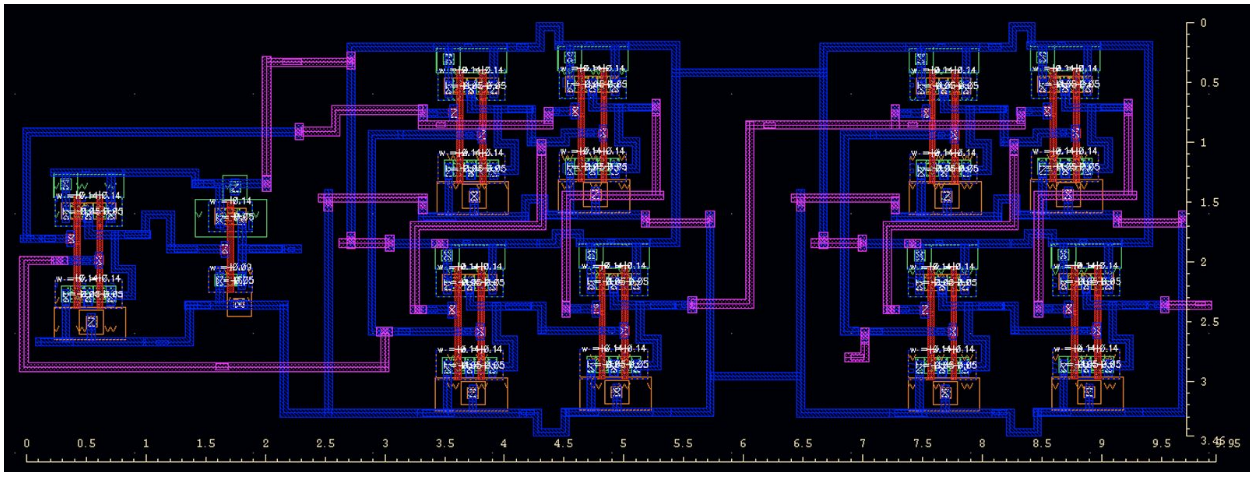
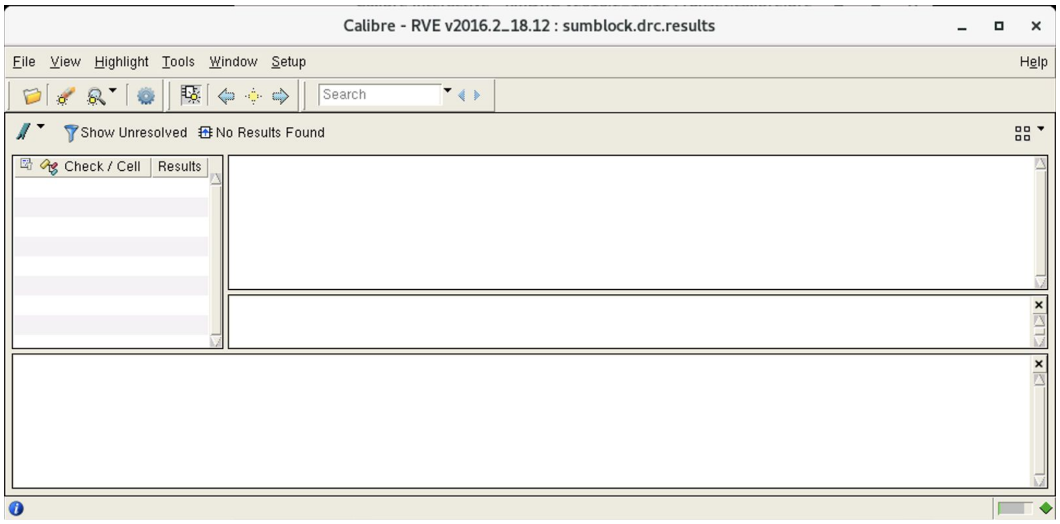


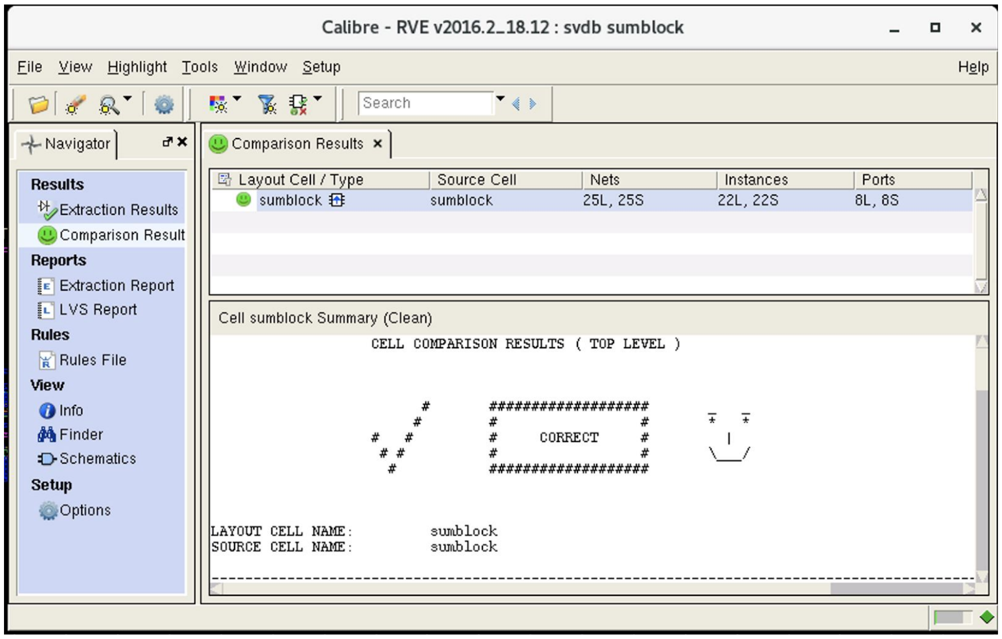
Figure 2 | Layout of the Sum Block



Calibre - RVE v2016.2_18.12 : sumblock.drc.results

Show Unresolved   No Results Found

Figure 2a | DRC Report of the Sum Block



Calibre - RVE v2016.2_18.12 : svdb sumblock

Layout Cell / Type	Source Cell	Nets	Instances	Ports
sumblock	sumblock	25L, 25S	22L, 22S	8L, 8S

Cell sumblock Summary (Clean)

```
                    CELL  COMPARISON  RESULTS  (  TOP  LEVEL  )



                          #        ###################
                         #         #                 #
                    #   #          #     CORRECT     #      _   _
                     # #           #                 #       \___/
                      #            ###################


LAYOUT CELL NAME:          sumblock
SOURCE CELL NAME:          sumblock

--------------------------------------------------------------------------------
```

Figure 2b | LVS Report of the Sum Block

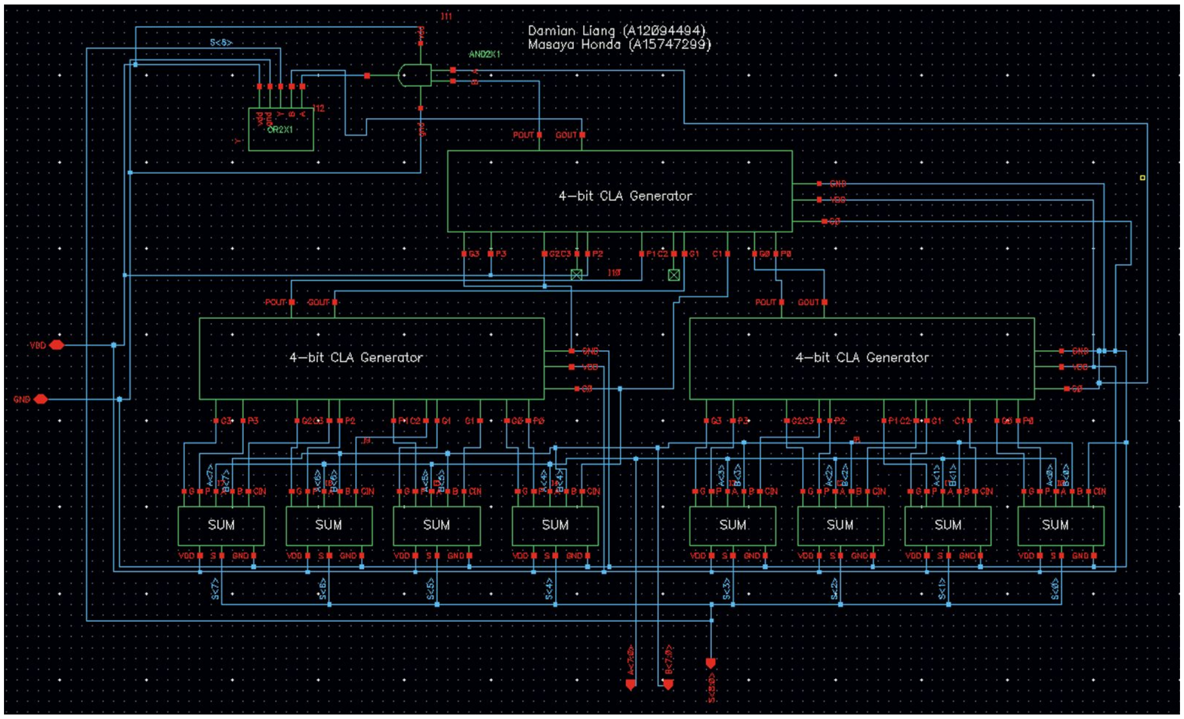


Figure 3 | Schematic of the 8-bit Custom Adder

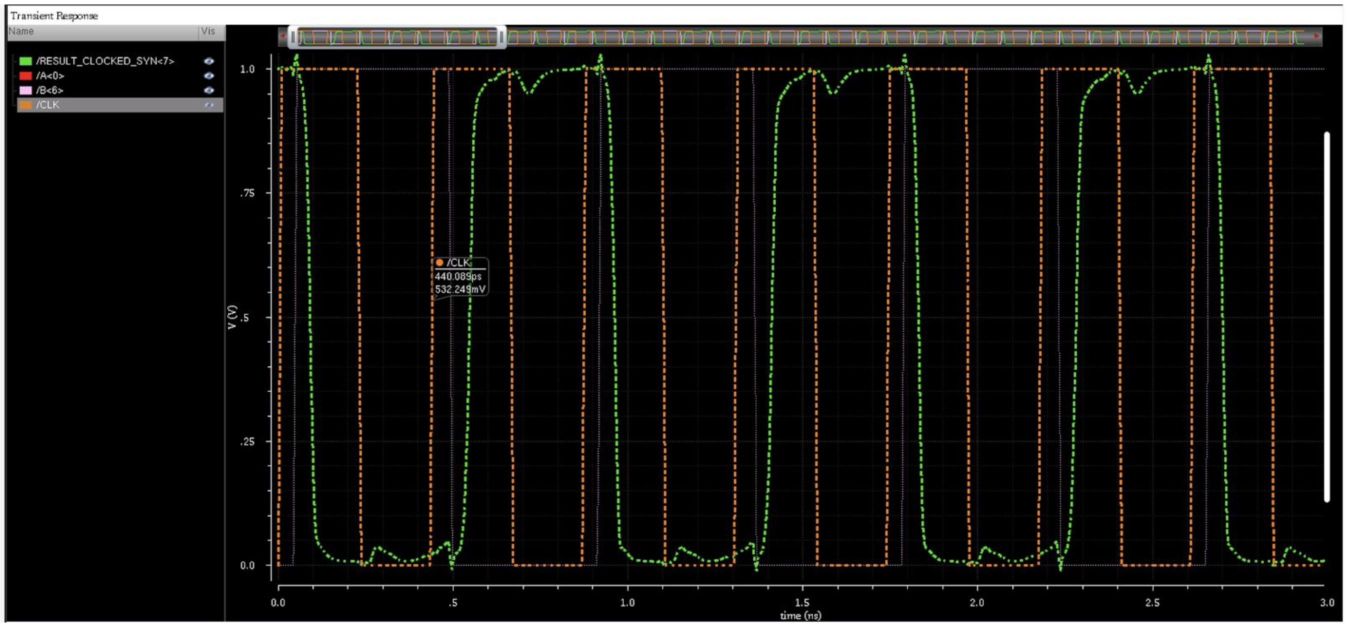


Figure 4a | Maximum Frequency for the Post-Extracted Lab 4 Adder

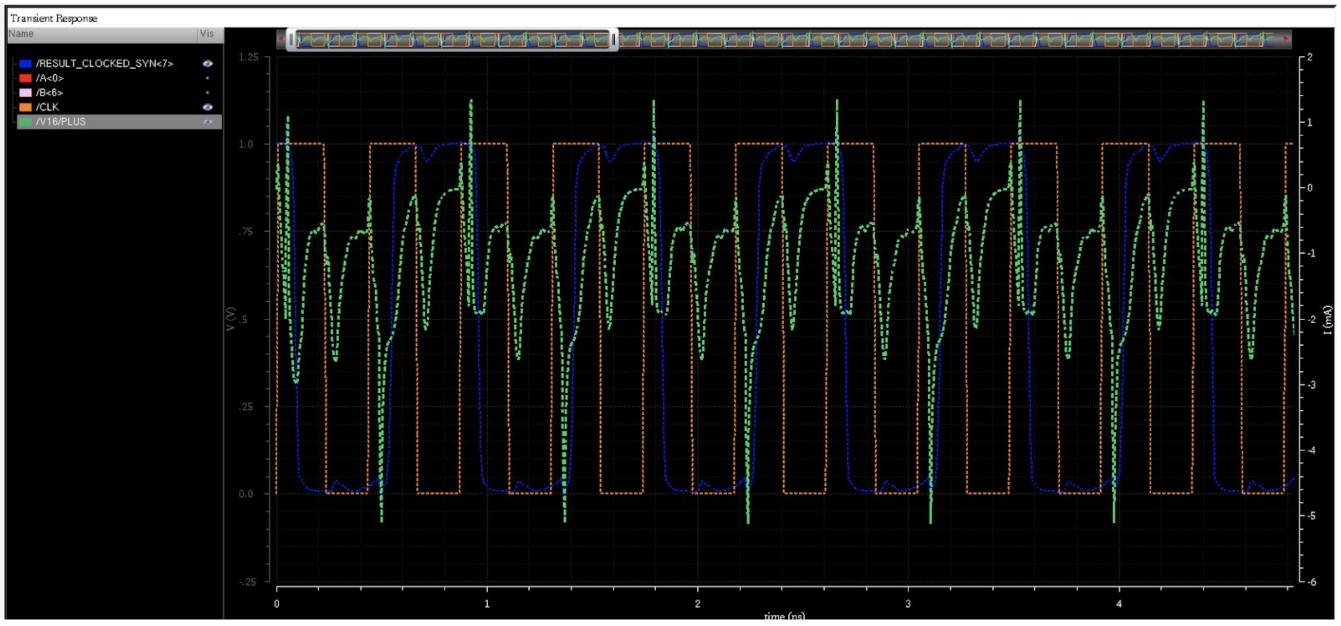


Figure 4b | Transient Current Waveform for the Post-Extracted Lab 4 Adder (498.349 ps to 15 ns)

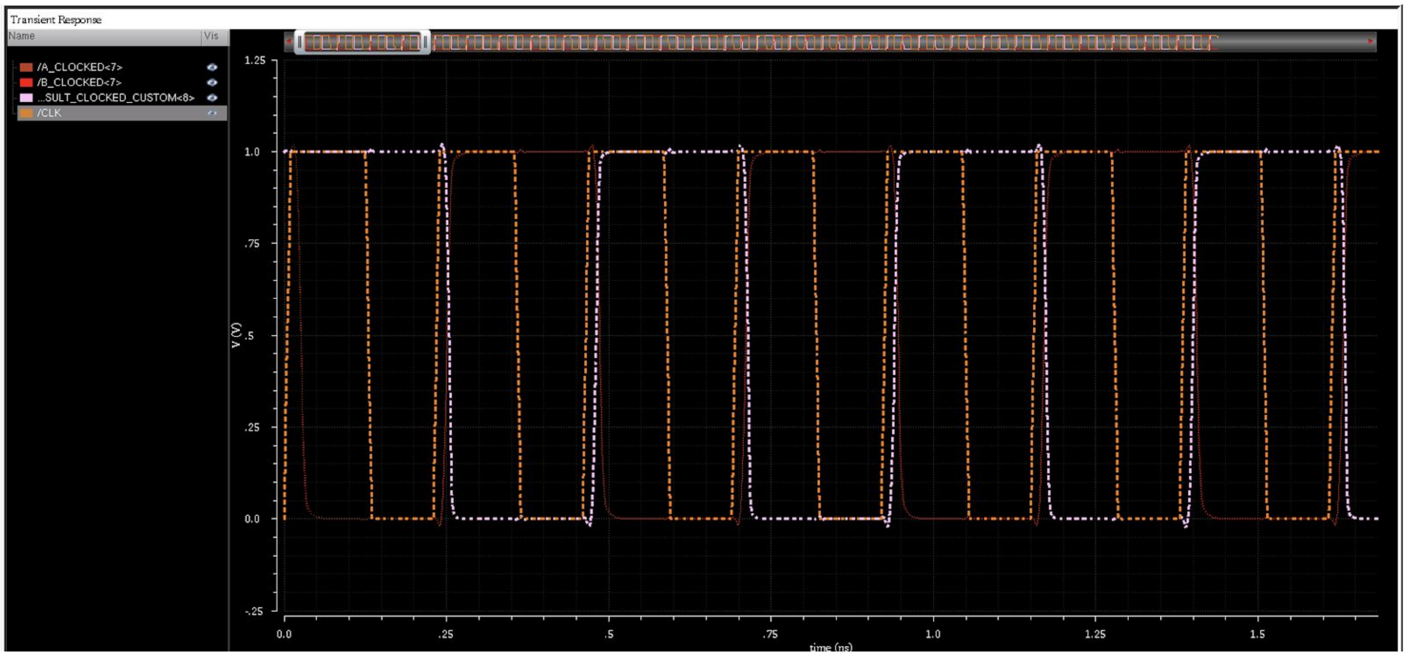


Figure 4c | Maximum Frequency for the Post-Extracted Custom Adder

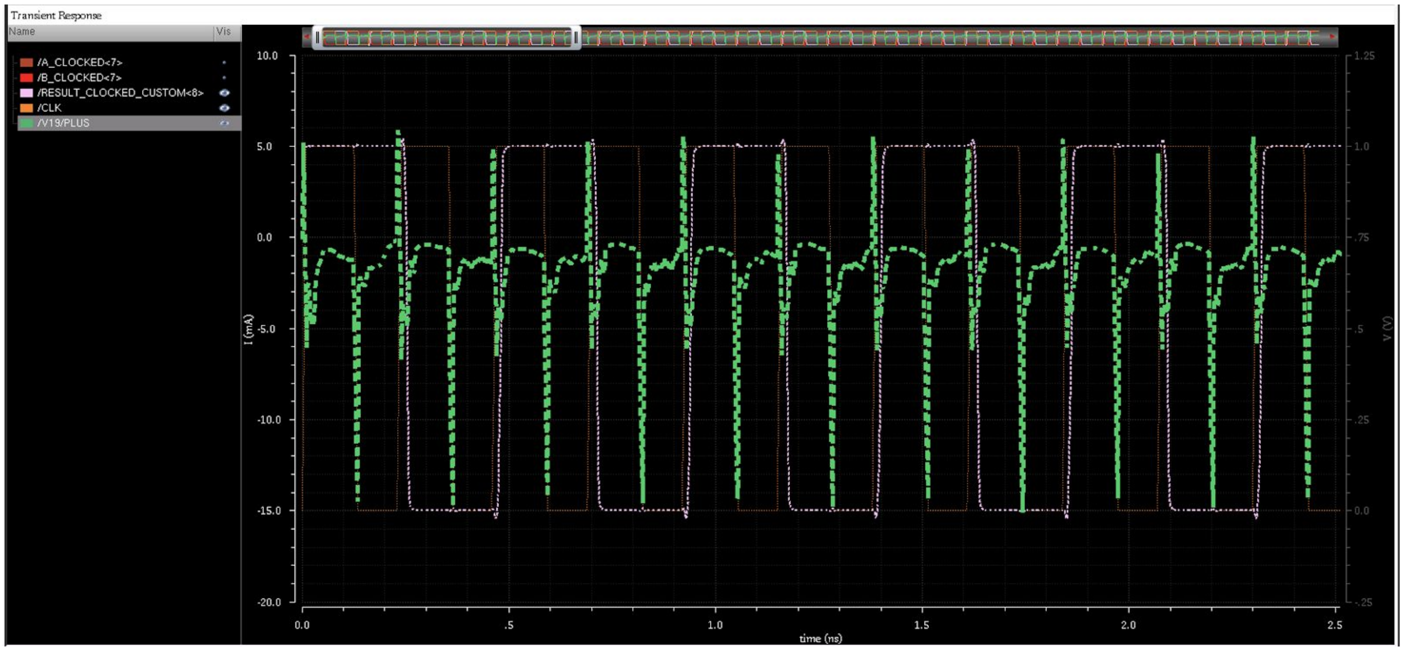


Figure 4d | Transient Current Waveform for the Post-Extracted Custom Adder (470.624 ps to 15 ns)

	Place+Route Schematic	Custom Design Schematic	Place+Route Extracted	Custom Design Extracted
	Performance for V_{DD} = 1.0V	Performance for V_{DD} = 1.0V	Performance for V_{DD} = 1.0V	Performance for V_{DD} = 1.0V
f_{max}	4.1 GHz	6.22 GHz	2.3 GHz	4.35 GHz
Power Consumption @ f_{max}	1.203E-3 W	2.11E-3 W	1.206E-3 W	1.891E-3 W
Energy per operation @ f_{max}	5.84E-13J/Cycle	6.83E-13 J/Cycle	10.5363E-13 J/Cycle	8.700E-13 J/Cycle
	Performance for V_{DD} = 1.0V, f_{CLK} = 1 GHz	Performance for V_{DD} = 1.0V, f_{CLK} = 1 GHz	Performance for V_{DD} = 1.0V, f_{CLK} = 1 GHz	Performance for V_{DD} = 1.0V, f_{CLK} = 1 GHz
Power Consumption @ 1GHz	421.1E-6 W	638.4E-6W	471.5E-6 W	659.6E-6 W
Energy per Operation @ 1GHz	8.422E-13 J/Cycle	9.430E-13 J/Cycle	1.319E-12 J/Cycle	1.277E-12 J/Cycle
	Other Important Parameters	Other Important Parameters	Other Important Parameters	Other Important Parameters
Adder Architecture	Ripple Carry	Hierarchical Carry-Lookahead Adder	Ripple-Carry	Hierarchical Carry-Lookahead Adder
Core Area	-	-	396.209 μm^2	34.427 μm^2
Critical Input Pair (i.e., A=?, B=?)	A=00000001 B=01111111	A=11111111 B=10000000	A=00000001 B=01111111	A=11111111 B=10000000
Transistor Types Used (e.g., VTL, VTG)	VTL	VTG	VTL	VTG

Figure 5 | Table of Results