Brent-Kung Adder

Nada Ali A16619058

Connor Kuczynski A15952472

The Brent-Kung adder is one of the parallel prefix adders which is a form of the carry-lookahead adder. We chose Brent-Kung because it has less wiring which leads to better performance and a smaller area. The advantages of the Brent-Kung adder is that it has minimum area and power compared to other parallel prefix adders. The disadvantage of this adder is that it doesn't fan-out as well as other prefix adders. Fan-out causes splitting and weakens the current that propagates through the adder. We mitigated that by adding buffers. Adders are used in most digital systems in today's world. However, some applications require low power and area like mobile phones, while server farms require high speed. [1]

Alternative Design Options

We have many ways to build lookahead trees. Three main trees are the Brent-Kung, Kogge-Stone, and Sklansky. Since we chose to make the Brent-Kung adder, we'll discuss the other two adders. For the Sklansky adder, it reduces the delay to $\log_2(N)$ stages, which causes fan-outs that are doubled at each level. These fan-outs cause the adders to perform poorly unless signals are buffered. For the Kogge-Stone adder, it reaches both $\log_2(N)$ stages and fan-out of 2 at each stage, which is good for speed, but it comes at the cost of having many long wires which causes it to have a bigger area if it's not in a regular grid. The Kogge-Stone also has a difficult layout with many parasitic capacitors.[1]

Since there are a lot more grey and black cells in the Kogge-Stone adder than there is in the Brent-Kung, it causes an increase in the power consumption, which makes it not be optimal for power. The same goes for the Sklansky adder. These topologies have larger area and power.[1]

Brent-Kung Topology

Our chosen architecture here is the Brent-Kung adder. We decided to use this architecture because it is optimal for power, unlike other architectures that were previously mentioned. According to the textbook, Brent-Kung adder has fewer wires and elements (black and grey cells), which gives us a smaller area and lower power consumption. We are designing for low power and reduced area which makes the Brent-Kung topology an attractive option. While being lower power compared to the Kogge-Stone, it still is capable of good performance (3GHz+) with optimizations on the critical path.

Most Innovative Parts of Design

The most innovative aspect of the define was using a reduced Vdd of .9V and optimal sizing of the pull-up and pull-down networks for the grey and black cells. We added black and grey cells to help in the generation and propagation process of a carry to the next stages. Black cells consist of one OR gate and two AND gates, and the grey cell consists of one AND gate and one OR gate. We used even and odd rows to minimize the inverting stages using the black and grey cells. For the even black and grey cells, we change their design by pushing in the bubbles in the inverted AND gates and turn them into a NOR gate. The same goes for the even grey cell where it's only one inverted AND gate that we change to being a NOR gate. This helps reduce the power consumption of the different stages while also reducing the number of stages needed. This allowed for lower Vdd and sizing while still having (3GHz+) performance.

The reason why we used lower threshold voltage gates was to mitigate the effect caused by the lower Vdd. The Vgs value is dependent on Vdd-Vt, if we decrease Vdd, we need to also decrease Vt to keep the same current through the gate. Because P=VI, by reducing the Vdd and keeping the current the same, we reduce power while also keeping the same performance as before. We sized the critical input gates larger so that there's less delay on the critical path. We did custom pull-up and pull-down networks for the gray and black cells so we can implement single-stage boolean expressions. We put the gate with the critical input closer to the output compared to the other transistors to reduce the delay from the critical input. We also sized the pull-up and pull-down networks for equal rise and fall time. Using pull-up and pull-down networks of the grey and black cells, we used smaller transistors compared to the built-in logic gates, which also decreases the current and therefore overall power.

Issues Faced in Implementation

Building the even and odd black cells to reduce the number of bubbles caused some issues implementing them in the adder. Creating test benches that tested each individual unit of the adder was time-consuming to run because each individual input had to be tested manually. This made it difficult to find where the error occurred while in development as each cell had to be tested manually every time. Ideally, in the future, we would be able to create a test script that would be able to quickly verify the individual units of the adder while iterating on the design.

Future Improvements

One of the things we could do is that we could taper the gates so that there's better drive strength and reduce the issue of the fan-out caused by the topology of the design. Another thing could be is to create a custom XOR gate that would allow for reduced number of transistors, and it could be properly sized for better speed compared to the build-it XOR gate. Propagate and generate for each bit could also be changed to use custom gates specifically a custom XOR and a custom AND gate. One of the things that we could also do is experiment skewing the sizing of the transistors on the critical path to respond to the critical input faster/better. If we had a better understanding of the speed and power requirements, we could try increasing or decreasing Vdd which would have the tradeoff of increasing the power and decreasing the speed or decreasing the power and increasing the speed. We could optimize Vdd to better match the power budget and speed requirements. Finally, if the adder requires high throughput, we could try using pipelining in between each stage which would have the drawback of high latency.

Connor designed, implemented, and tested the adder. Nada compiled the images and report together.

Acknowledgments:
We would like to thank Reilly Jensen for his help in this project, as well as the rest of the teaching team.

References:
[1] CMOS VLSI Design: A Circuits and Systems Perspective
[2] Professor Patrick Mercier's lecture notes.

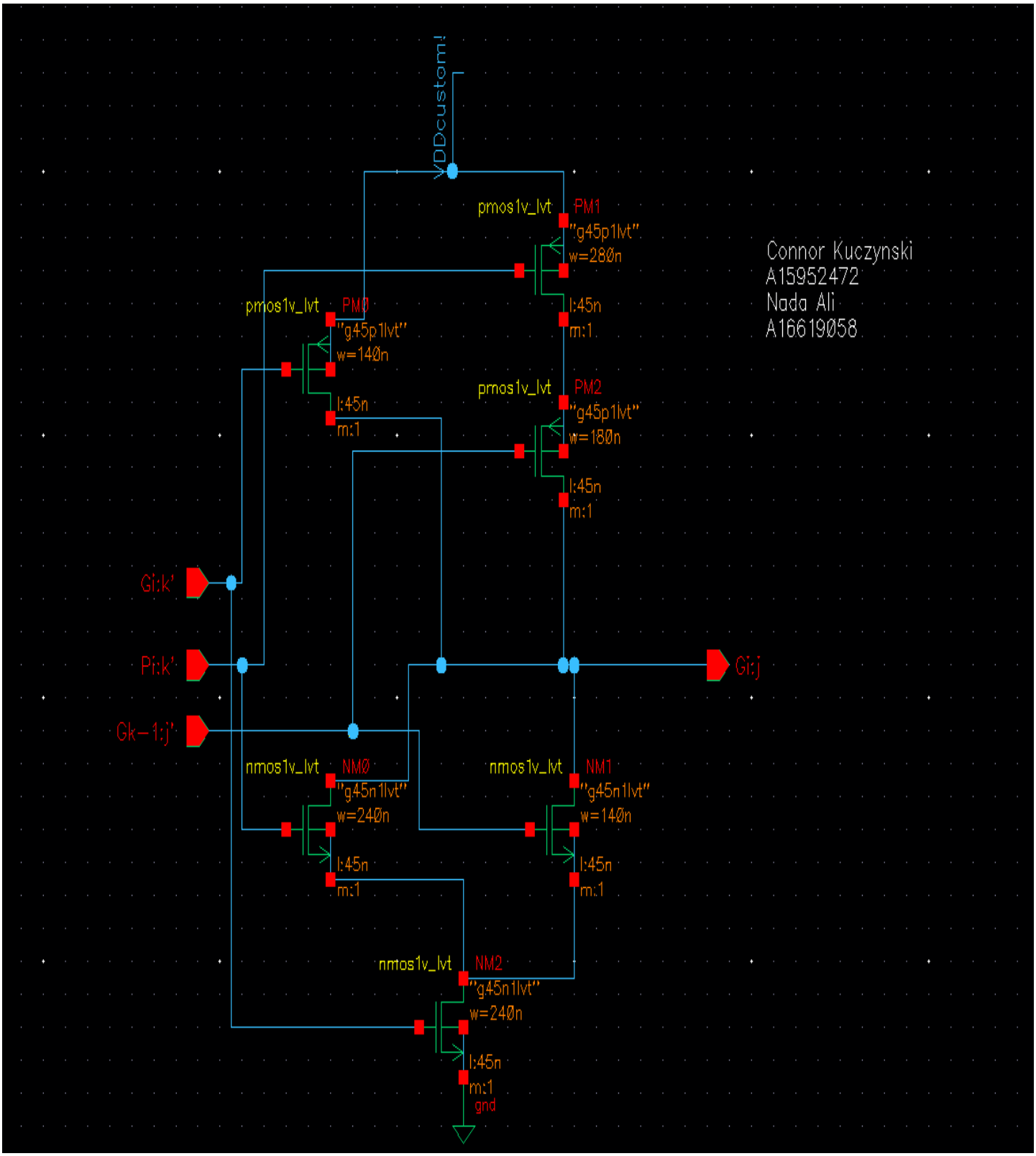

Figure 1 | Grey_cell_odd- Largest individual cell on the critical path.

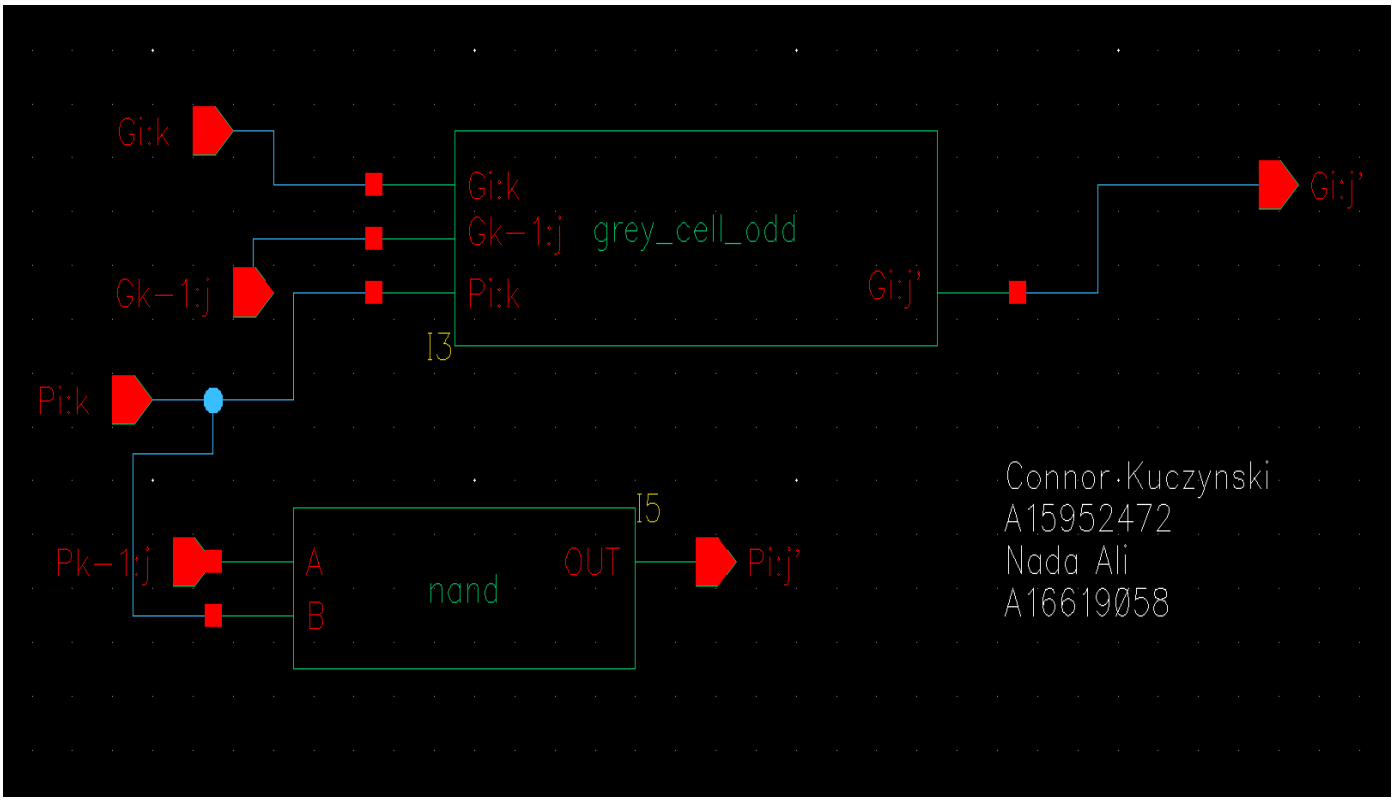

Figure 2 | Black Cell

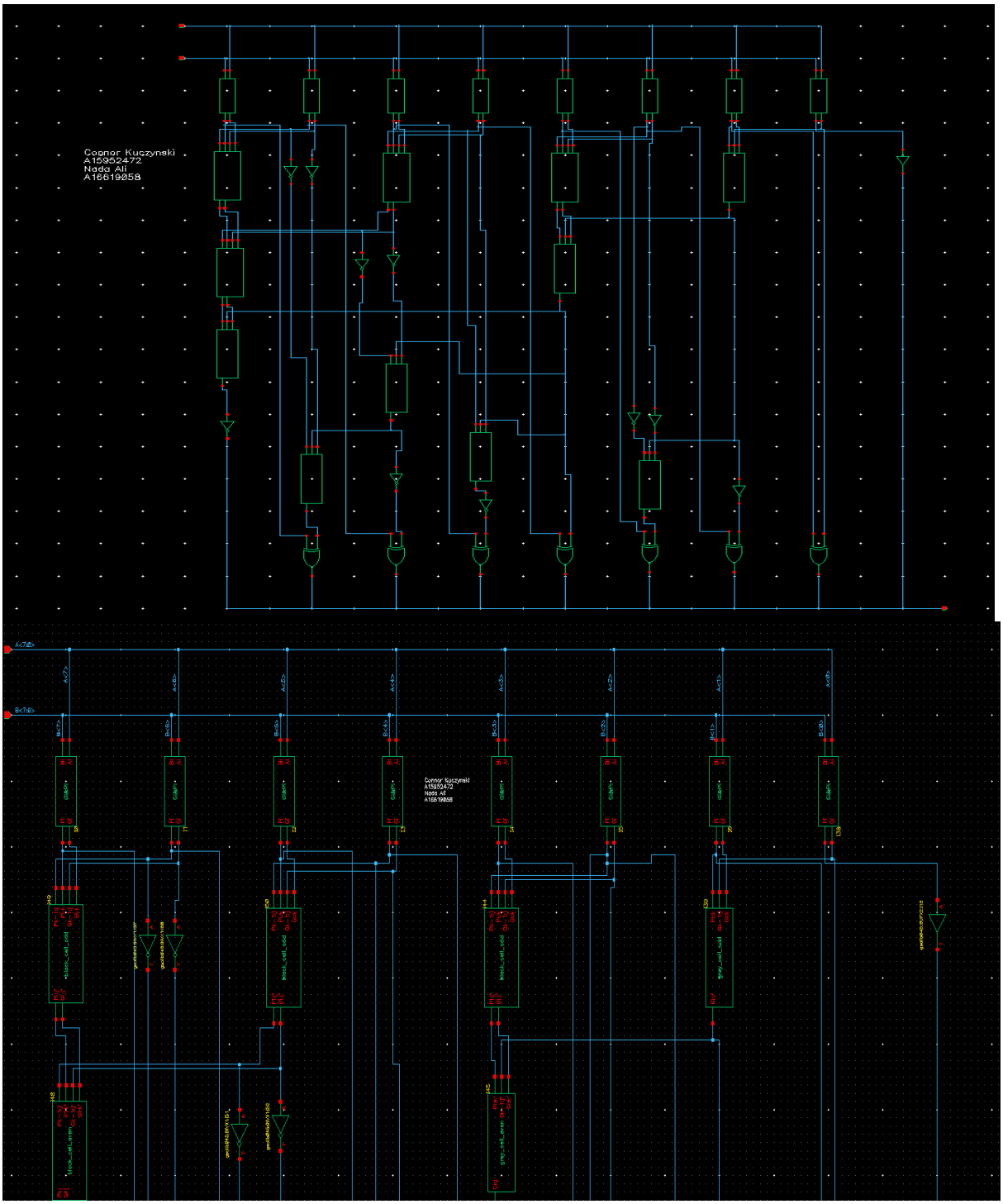

Figure 3 | Schematic of 8-bit Adder, A zoomed-in version of the adder.

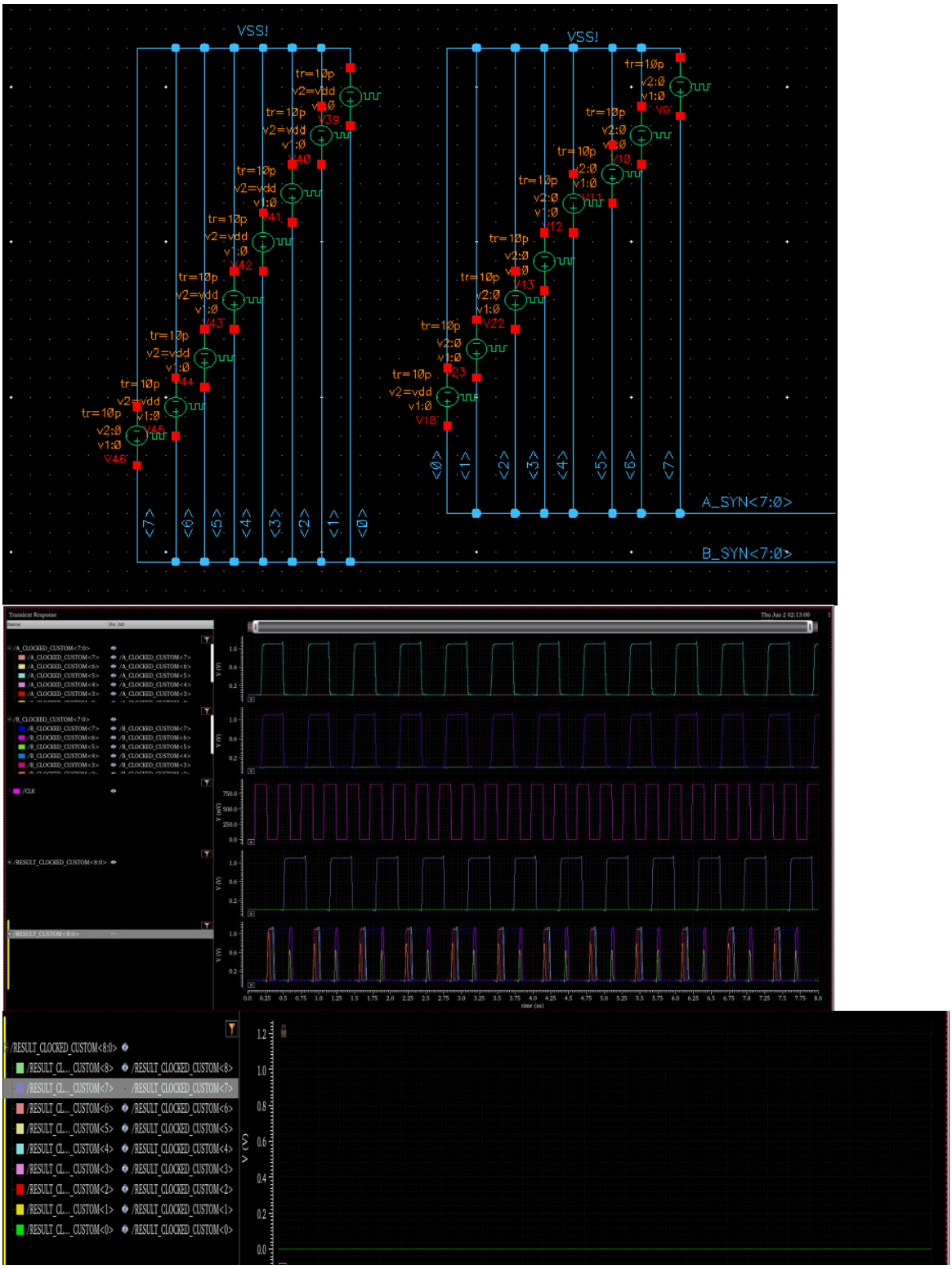

Figure 5.1 | Critical Input, Custom Critical Input. Proof of Correctness for Custom Adder

Figure 5.2 | Synthesized Critical Input. Proof of Correctness for Synthesized Adder

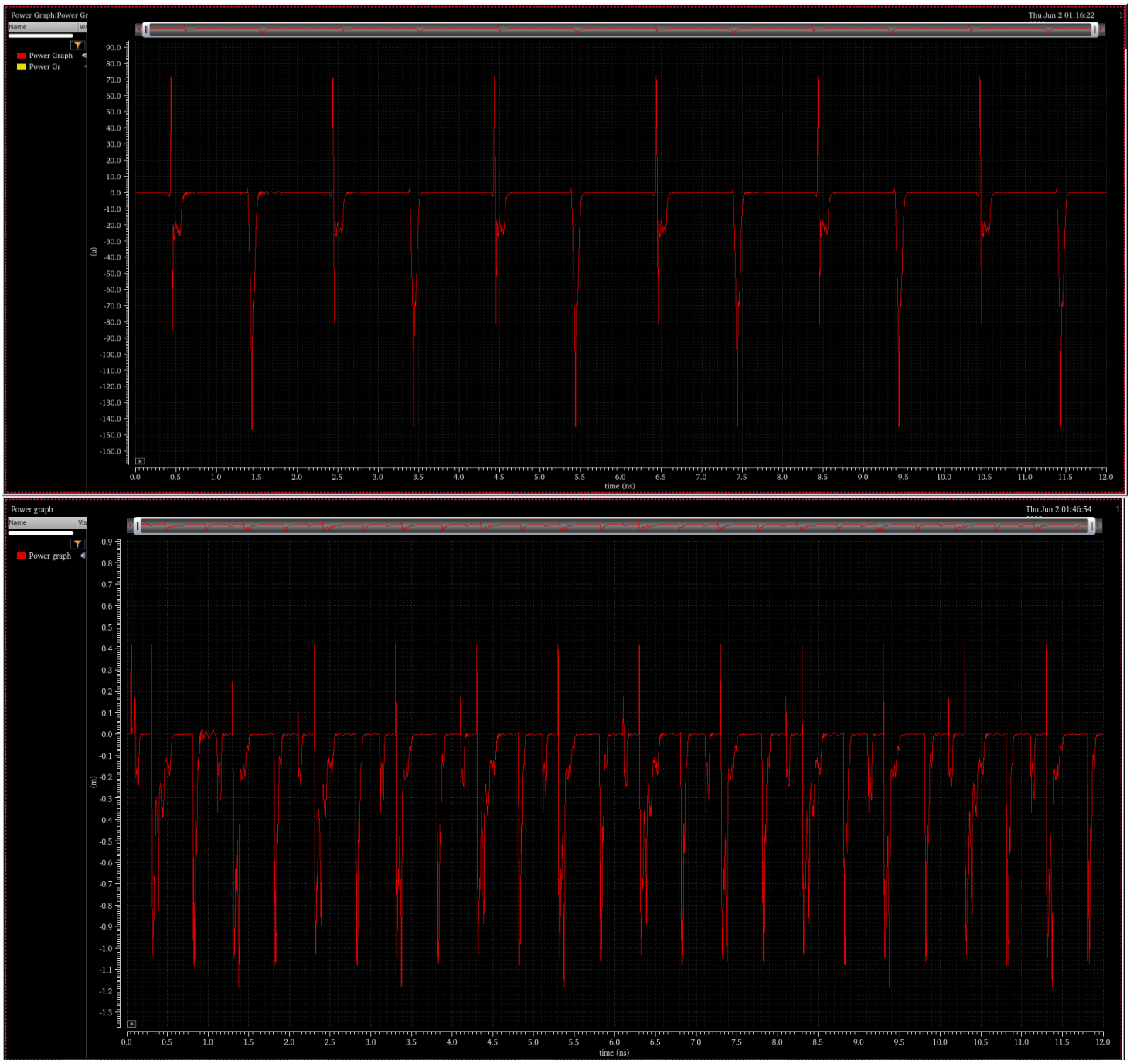

Figure 5.3 | Custom (top) and Synthesized (bottom) Power at Graphs at 1GHz

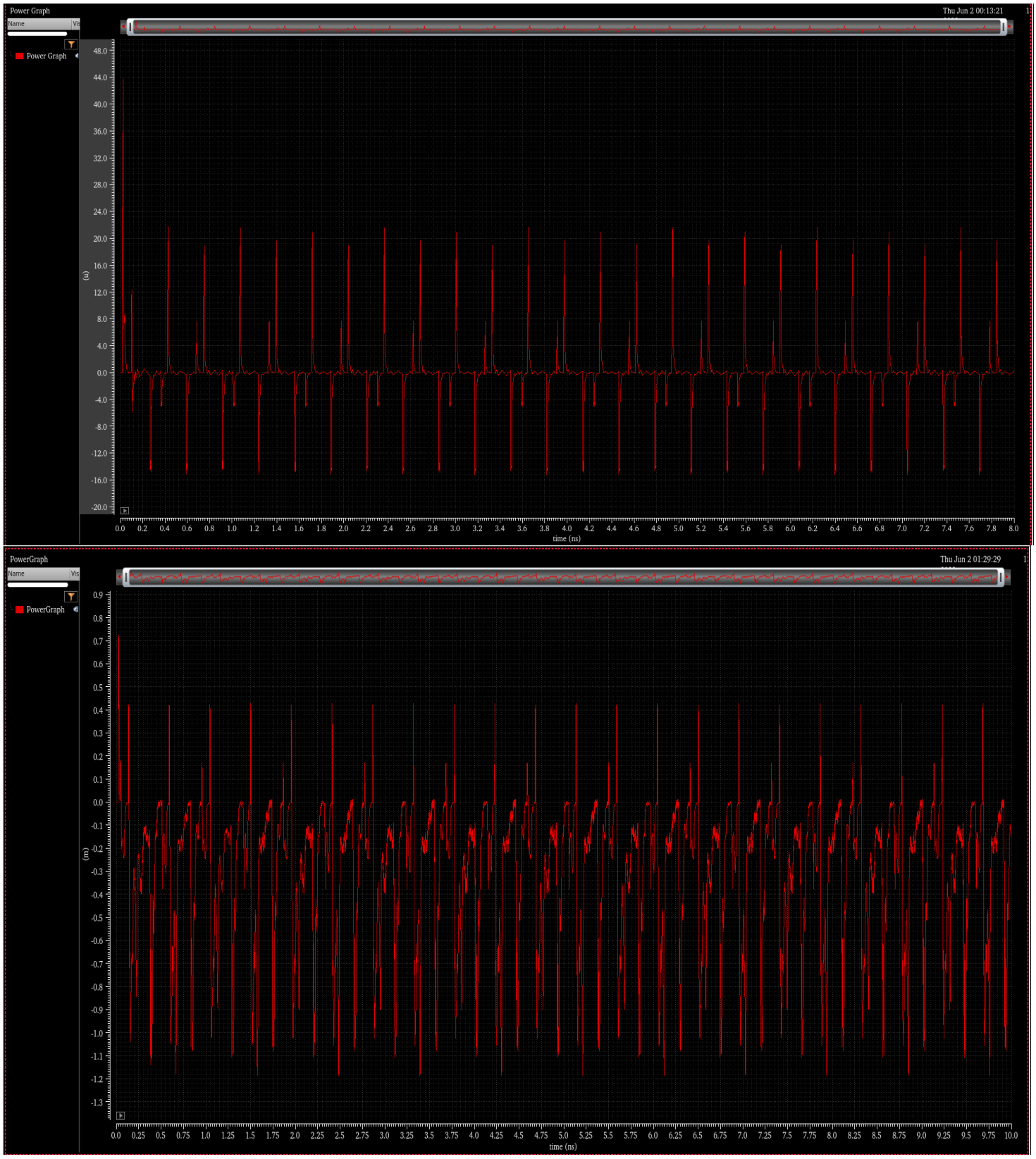

Figure 5.3| Custom (top) and Synthesized (bottom) Power at Graphs at 3.1GHz and 2.2GHz respectively.

	Place+Route Schematic	Custom design schematic
	VDD = 1.1V	VDD = .9V
f_{max}	2.2GHz	3.1GHz
Power consumption @ f_{max}	154uW	391uW
Energy per operation @ f_{max}	70 fJ / cycle	126 fJ / cycle
	VDD = 1.1V f_{clk} = 1GHz	VDD = .9V 1GHz
Power consumption @ 1GHz	133uW	120uW
Energy per operation @ 1GHz	133 fJ / cycle	120 fJ / cycle
Adder architecture	Ripple-carry	Brent-Kung
Critical Input Pair (A=?,B=?)	A = 0111_1111 B = 0000_0000	A = 0111_1111 B = 0000_0001
Transistor types used	VTG	VTL

Time clipped: 12.04ns 15.268 ns (10 clock cycles at 3.1GHz)

Time clipped: 12ns 22ns (10 clock cycles at 1GHz)

Time clipped: 12ns 16.54ns (10 clock cycles at 2.2GHz)

Figure 6 | Table of Results