8-Bit Carry Lookahead Adder

Thao Nguyen A15409684
Shane Minerman A15505071

This project features a hand-design of an 8-bit adder schematic that has faster performance than the adder we synthesized in Lab 4. The adder has inputs A<7:0> and B<7:0> that will arrive from the outputs of positive-edge triggered flip-flops and outputs S<8:0> that will be loaded by the D input of positive-edge triggered flip-flops. The chosen adder architecture should not be the standard simple ripple-carry design and the full adder itself should be a custom, transistor-level cell. Binary adders are important because they are used in arithmetic logic units (ALUs) of many computers and other kinds of processors to increment and decrement operators, calculate addresses, table indices, and other similar operations. Specifically, 8-bit microcontrollers are commonly used in home devices and other low power and cost applications that require integrated high performance features. Some examples include smoke detectors, thermostats, microwaves, and washing machines.

The basic full adder follows the logic equations $S = A \oplus B \oplus C = \overline{B}A\overline{C} + \overline{A}B\overline{C} + \overline{A}C\overline{B} + ABC$ and $Cout = AB + C(A + B)$. The transistor implementation of the sum and carry out circuits use mirrored pull-up and pull-down networks due to function symmetry, so the design consists of 32 transistors overall (including inverters). Additionally, the simple ripple-carry architecture with cascaded full adders is highly nonoptimal because the critical path goes from Cin to Cout, so in most cases the carry has to propagate through the entire network to produce the correct output.

Optimal adder design aims to reduce propagation delay on the critical path. Our chosen implementation for the full adder uses boolean algebra to transform the sum function:

$$\begin{aligned} S &= \overline{B}A\overline{C} + \overline{A}B\overline{C} + \overline{A}C\overline{B} + ABC) \\ &= ABC + (A + B + C)(\overline{A}\,\overline{B} + \overline{A}\,\overline{C} + \overline{B}\,\overline{C}) \\ &= ABC + (A + B + C)\ \overline{Cout} \end{aligned}$$

In the above implementation, the Cout term is shared in the sum. Cout is on the critical path, so the goal is to compute it first and then utilize it in the sum circuit to reduce worst case propagation delay–even if it takes longer to generate the sum, it should not significantly affect the critical path. This design is known as the mirror adder (*fig. 1)* in which $\overline{Cout}$ is computed and taken as the input to the sum function. The total size of this design is 28 transistors (including inverters).

The sizing of this mirror adder focuses primarily on the critical path from Cin to Cout. The Cin transistors are placed closest to the output because their inputs will arrive last, thus the A and B transistors will have already charged or discharged their capacitances, leading to reduced propagation delay. The Cin path should be larger than minimum size for speed reasons and is consequently sized according to the PDK PMOS:NMOS ratio 7/6:1. Therefore, the PMOS's on the Cin path are size 7/3 and the NMOS's are size 2. All other transistors not on the critical path are set to be minimum size. This means the entirety of the sum circuit is size 1, along with the four transistors that are in the Cin to Cout circuit but are not directly on the Cin path.

The chosen architecture is the carry-lookahead adder. This method uses generate (G) and propagate (P) logic to reduce propagation delay. Ripple-carry adders have to wait for the carry of each bit pair to be computed, resulting in a long output time. On the other hand, $G_i = A_iB_i$ and $P_i = A_i \oplus B_i$ only depend on inputs A and B, so G/P can be computed for all the bits in parallel. Using the carry equation $C_{i+1} = G_i + P_iC_i$ with recursive logic design, a carry-lookahead architecture is set up in which all the carries are computed in parallel and inputted into their respective full adders. The sum calculation does not have to wait for carry propagation, greatly reducing delay time.

The 8-bit carry-lookahead implementation was split into two 4-bit blocks (*fig. 2)*. The gates used are from the Cadence 45 nm standard cell library gsclib045. The 4-bit block takes input Cin and generates C1, C2, C3, and C4. Output C4 is then taken as the input to the next 4-bit block that generates C5, C6, C7, and C8, which is the Cout of the sum. The full 8-bit adder features eight full adder cells and two 4-bit carry-lookahead blocks (*fig. 3)*. One thing to note in the schematic is that all the Cout pins are grounded. This is because the nature of the architecture renders the need for carry-outs obsolete since carry-ins are simply generated by G/P logic. The sum only needs $\overline{Cout}$ in its computation, therefore the inverter for Cout can be taken out of the design to save space. However, we decided to include the additional transistors and the Cout pin for the wholeness of the full adder design.

The testbench is set up such that inputs A<7:0> and B<7:0> are generated from pulsed voltage waveforms and clocked through positive-edge triggered flip-flops. Output S<8:0> is also similarly clocked. The critical inputs are A = 0000 0001 and B = 0111 1111. Running transient simulations at VDD = 1.1 V for the synthesized lab 4 adder, it was found that the maximum frequency for the correct output is 2.1 GHz (*fig. 4a)*. To find power consumption of the adder and all the flip flops used, current was averaged over 10 cycles. The power at maximum frequency for the synthesized adder is 455.5 uW, and the energy per operation (= power*period) is 2.169E-13 J. Transient simulations for the custom adder resulted in a maximum frequency of 2.3 GHz, which is faster than the synthesized design (*fig. 4b)*. Measurements pertaining to power consumption, energy per operation, and other important parameters for both designs are listed in figure 5.

Many of the challenges we encountered in this design project were related to issues with VDD and VSS levels. For example, in the design of the carry-lookahead block the high and low output voltages did not hit their expected values. We found this was because the premade gates used from the library required set up of global nets, a method that we were unfamiliar with. There were power and ground rails on our top level schematic for other components, but the gates were essentially left floating. This error was resolved with helpful global net documentation.

This section proposes future design iterations for better performance. The current full adder design uses four additional transistors for two inverters, but we can reduce inverting stages by utilizing the inversion property. Because the full adder computes $\overline{Cout}$, we can cascade a complementary full adder that uses complemented inputs to compute the complemented sum. Therefore, we do not need to add an extra inverter in the critical path to convert $\overline{Cin}$ to Cin for use in the next stage. This design consists of cascading alternating regular full adder (even) cells and complemented (odd) cells to save inverting stages and reduce critical path delay.

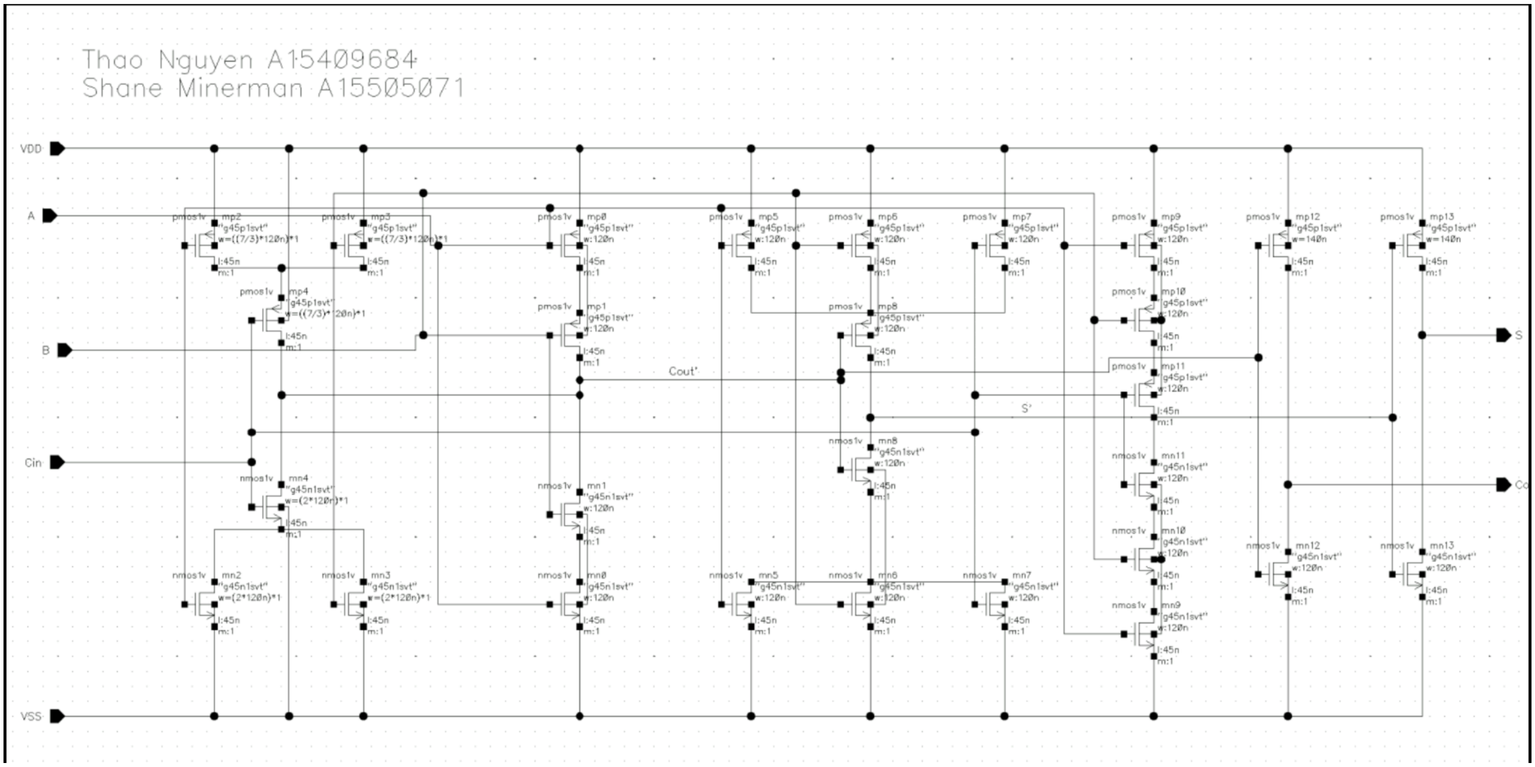


Figure 1 | 1-bit Mirror Adder Schematic

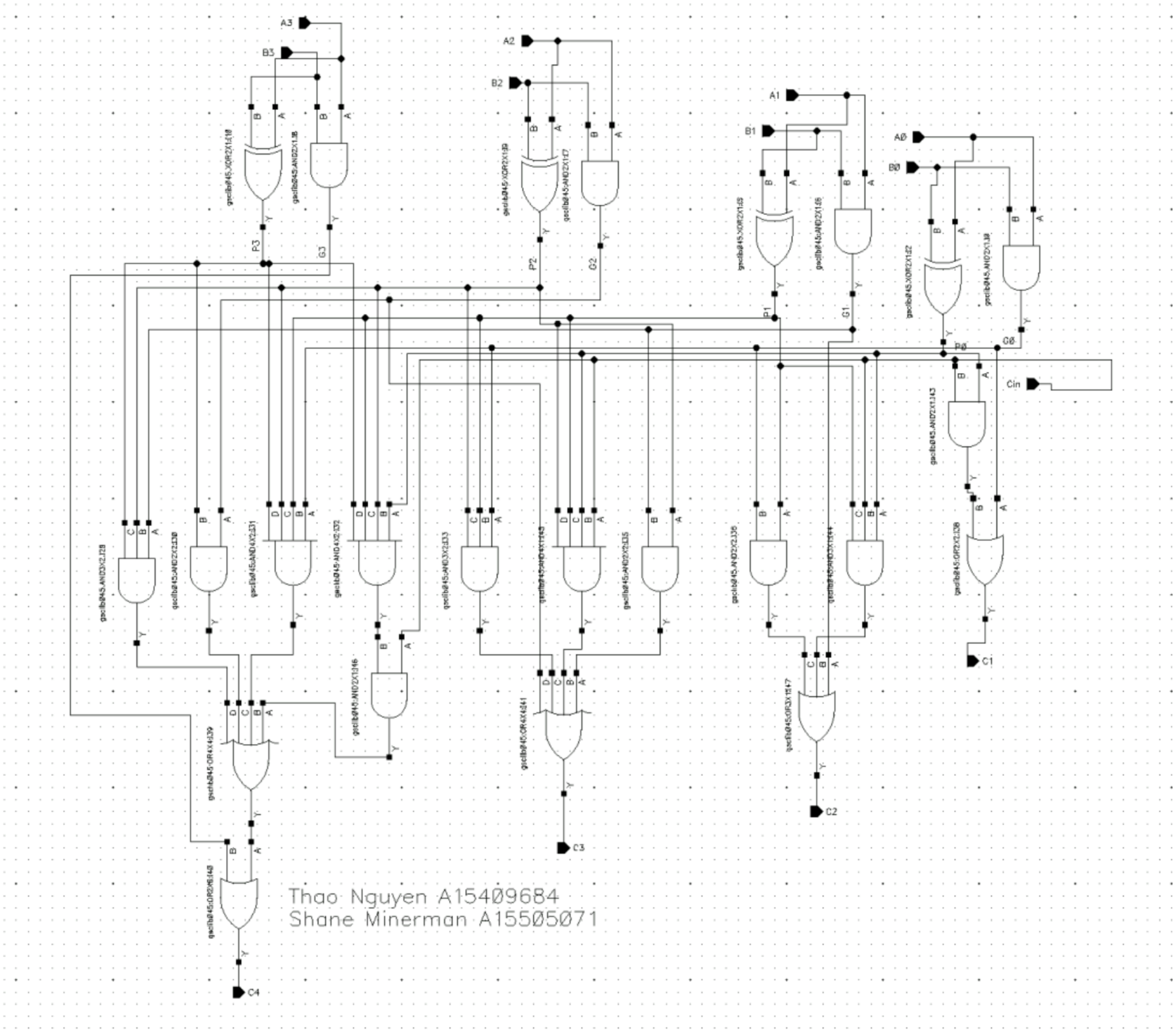


Figure 2 | 4-bit Carry Lookahead Block Schematic

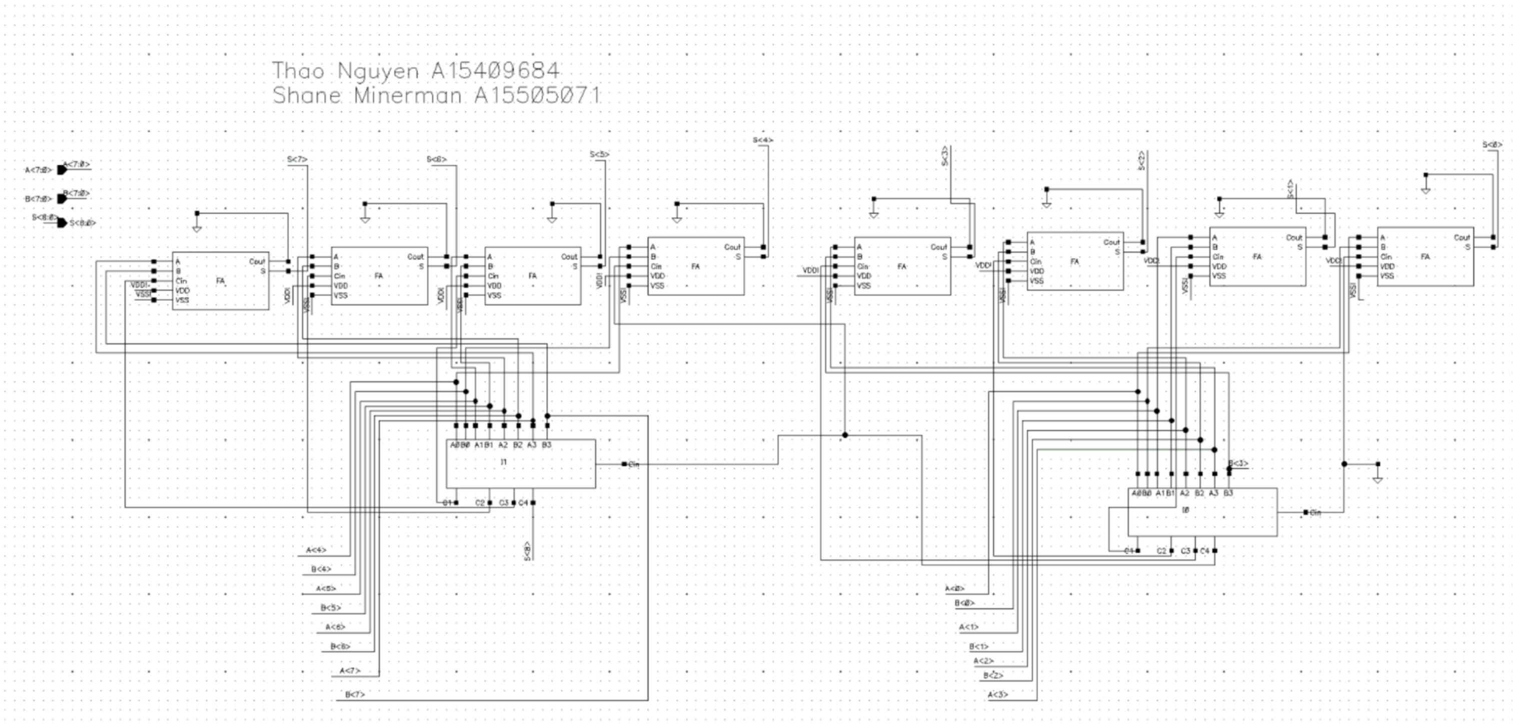
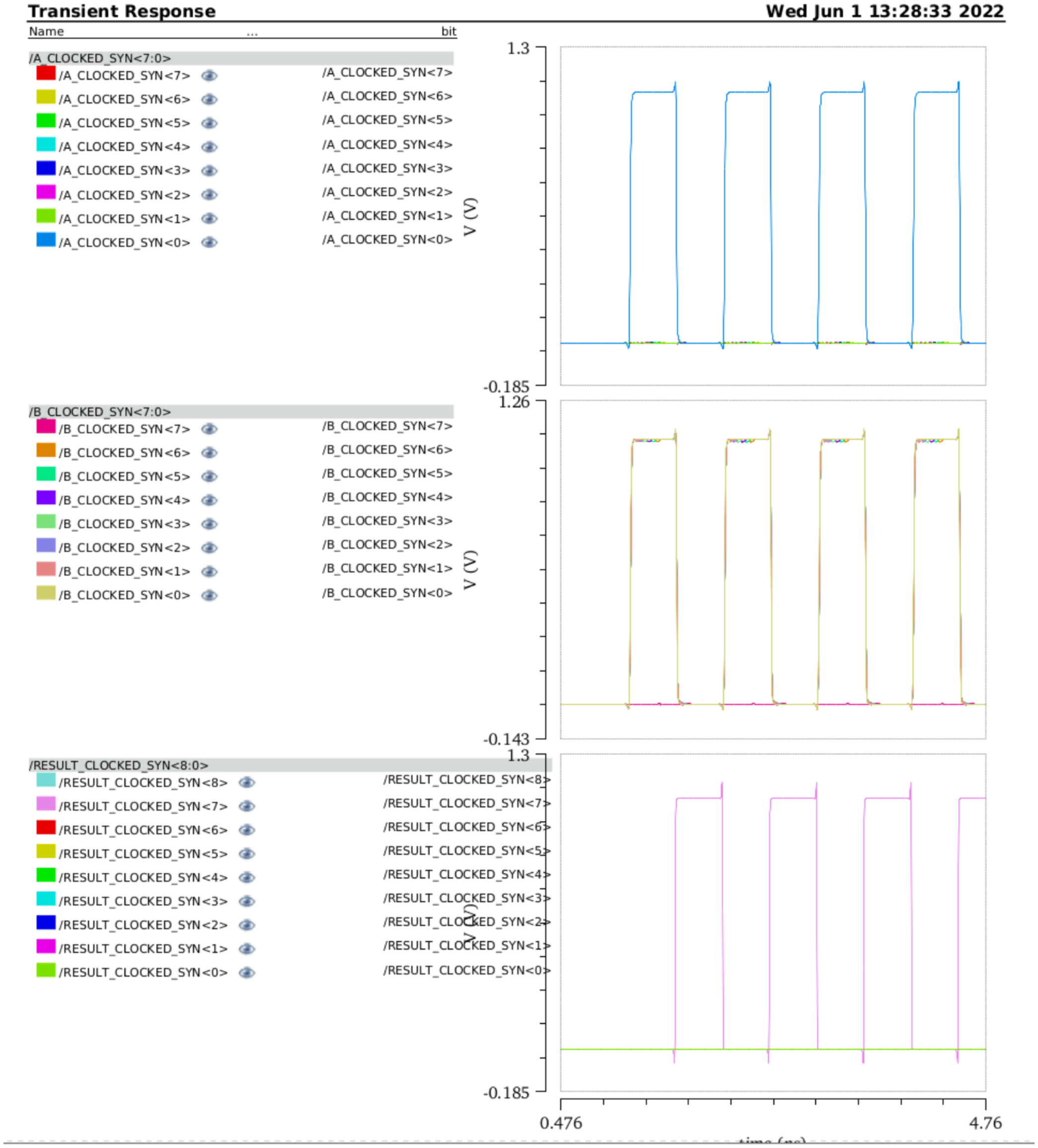


Figure 3 | 8-bit Adder Schematic



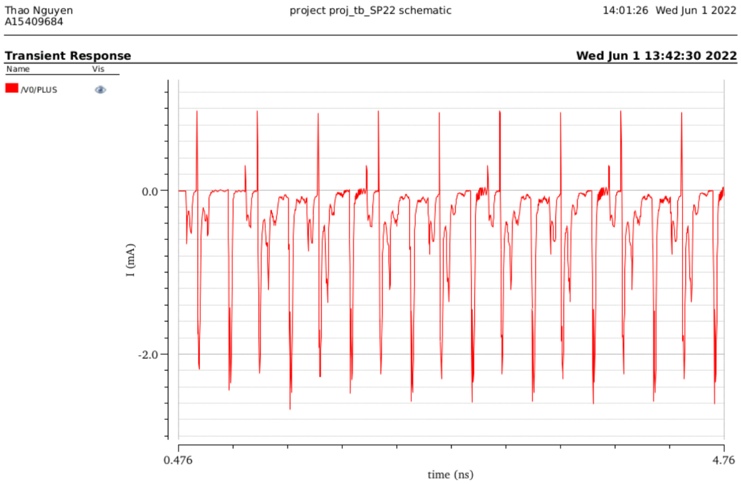
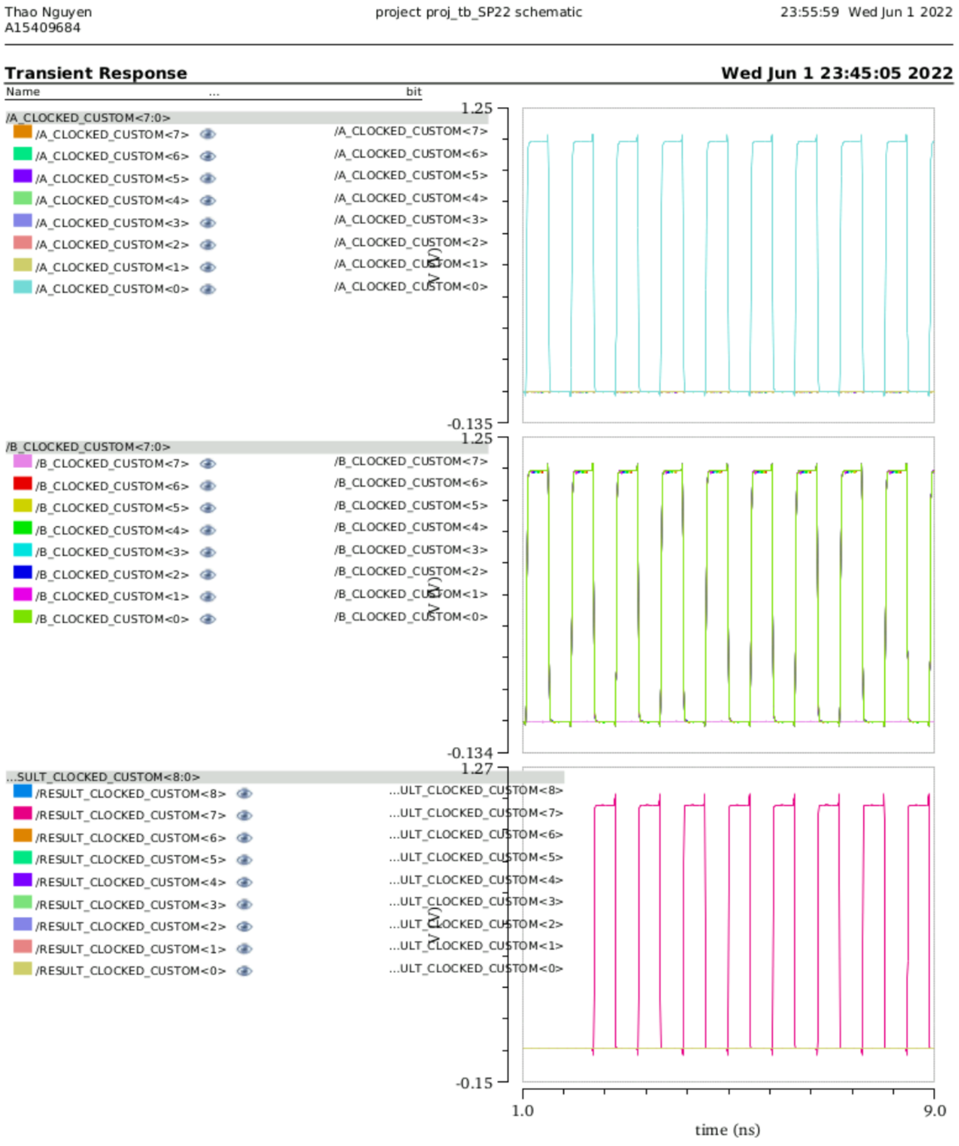


Figure 4a | Transient simulations of the synthesized adder. Graphs (top to bottom): A = 0000 0001, B = 0111 1111, S<8:0>, and system current. The maximum clock frequency is 2.1 GHz. Period is from 0.476 ns to 4.76 ns (10 cycles).



Thao Nguyen
A15409684

project proj_tb_SP22 schematic

23:55:59 Wed Jun 1 2022

Transient Response **Wed Jun 1 23:45:05 2022**

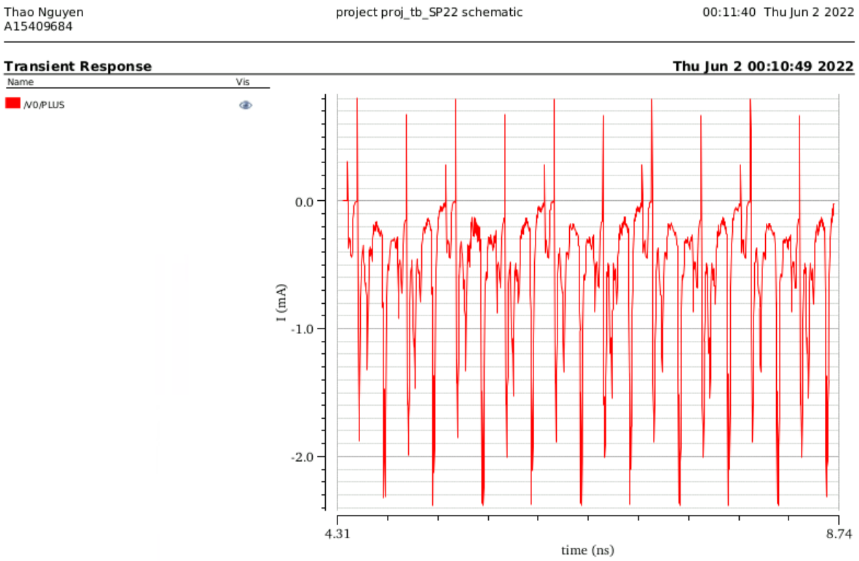
Name ... bit

/A_CLOCKED_CUSTOM<7:0>
- 🟧 /A_CLOCKED_CUSTOM<7>
- 🟩 /A_CLOCKED_CUSTOM<6>
- 🟪 /A_CLOCKED_CUSTOM<5>
- 🟩 /A_CLOCKED_CUSTOM<4>
- 🟦 /A_CLOCKED_CUSTOM<3>
- 🟥 /A_CLOCKED_CUSTOM<2>
- 🟨 /A_CLOCKED_CUSTOM<1>
- 🟦 /A_CLOCKED_CUSTOM<0>

/B_CLOCKED_CUSTOM<7:0>
- 🟪 /B_CLOCKED_CUSTOM<7>
- 🟥 /B_CLOCKED_CUSTOM<6>
- 🟨 /B_CLOCKED_CUSTOM<5>
- 🟩 /B_CLOCKED_CUSTOM<4>
- 🟦 /B_CLOCKED_CUSTOM<3>
- 🟦 /B_CLOCKED_CUSTOM<2>
- 🟪 /B_CLOCKED_CUSTOM<1>
- 🟩 /B_CLOCKED_CUSTOM<0>

...SULT_CLOCKED_CUSTOM<8:0>
- 🟦 /RESULT_CLOCKED_CUSTOM<8>
- 🟥 /RESULT_CLOCKED_CUSTOM<7>
- 🟧 /RESULT_CLOCKED_CUSTOM<6>
- 🟩 /RESULT_CLOCKED_CUSTOM<5>
- 🟪 /RESULT_CLOCKED_CUSTOM<4>
- 🟩 /RESULT_CLOCKED_CUSTOM<3>
- 🟦 /RESULT_CLOCKED_CUSTOM<2>
- 🟥 /RESULT_CLOCKED_CUSTOM<1>
- 🟨 /RESULT_CLOCKED_CUSTOM<0>

Printed on
by ee165sp22aq



Thao Nguyen
A15409684

project proj_tb_SP22 schematic

00:11:40 Thu Jun 2 2022

Transient Response **Thu Jun 2 00:10:49 2022**

Name Vis

- 🟥 /V0/PLUS

Figure 4b | Transient simulations of the custom adder. Graphs (top to bottom): A = 0000 0001, B = 0111 1111, S<8:0>, and system current. The maximum clock frequency is 2.3 GHz.The period for average power taken is 4.35 ns to 8.7 ns (10 cycles).

Performance for VDD = 1.1V

	Place + route schematic	**Custom design schematic**
fmax	2.1 GHz	2.3GHz
power consumption @ fmax	455.5 uW	572.7uW
energy per operation @ fmax	2.169E-13 J	2.49E-13 J

Performance for VDD = 1.1V, fclk = 1 GHz

	Place + route schematic	**Custom design schematic**
power consumption @ 1 GHz	201.3 uW	247.1 uW
energy per operation @ 1 GHz	2.013E-13 J	2.47E-13 J

Other important parameters

Adder architecture	Carry-ripple	Carry-Lookahead

Critical input	A = 0000 0001
	B = 0111 1111
	S = 1000 0000
	Cout = 0
		A = 0000 0001
		B = 0111 1111
		S = 1000 0000
		Cout = 0

Figure 5 | Table of Results