<u>Modules:</u>

64-bit CSA:

module CSA_64b(c_out, sum, a, b, c_in):

- This module is made up of 3 32-bit RCAs. It multiplies two inputs, each up to 64 bits long.

module FA_32b(c_out, sum, a, b, c_in):

- This module is made up of 2 16-bit RCAs. It adds two inputs, each up to 32 bits long.

module FA_16b(c_out, sum, a, b, c_in):

- This module is made up of 4 4-bit RCAs. It adds two inputs, each up to 16 bits long.

module FA_4b(c_out, sum, a, b, c_in):

- This module is made up of 4 1-bit FAs. It adds two inputs, each up to 4 bits long.

module FA_str(c_out, sum, a, b, c_in):

- This module is made up of 2 1-bit HAs. It adds two inputs, each 1 bit.

64-bit RCA:

module FA_64b(c_out, sum, a, b, c_in):

- This module is made up of 4 16-bit RCAs. It adds two inputs, each up to 64 bits long.

module FA_16b(c_out, sum, a, b, c_in):

- This module is made up of 4 4-bit RCAs. It adds two inputs, each up to 16 bits long.

module FA_4b(c_out, sum, a, b, c_in):

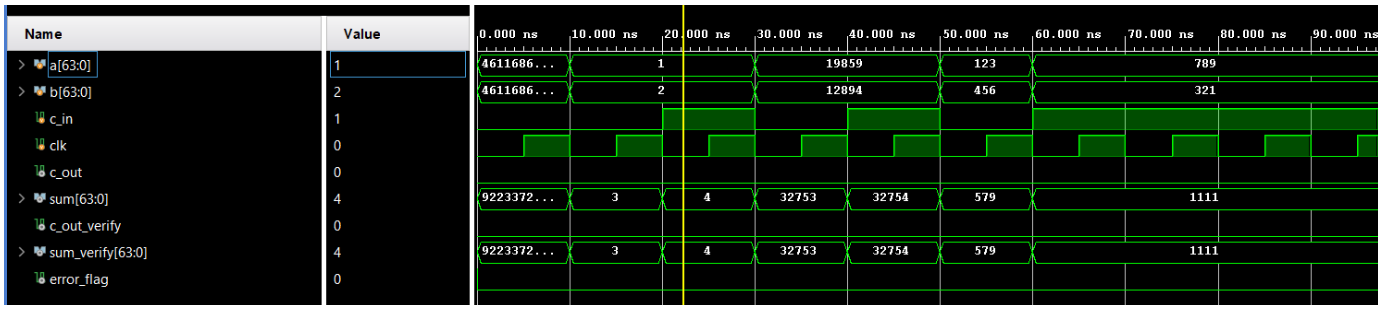
- This module is made up of 4 1-bit FAs. It adds two inputs, each up to 4 bits long.

module FA_str(c_out, sum, a, b, c_in):

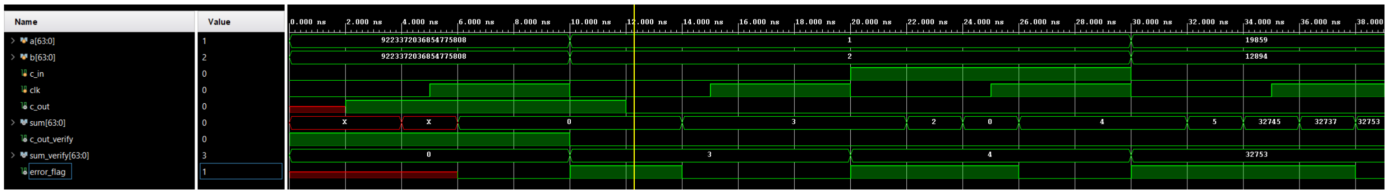
- This module is made up of 2 1-bit HAs. It adds two inputs, each 1 bit.

<u>Waveforms:</u>

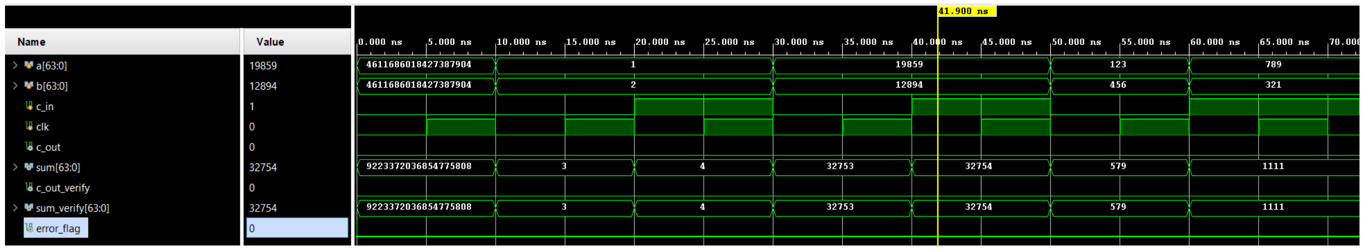
64-bit CSA:



64-bit CSA with 2ns Delay (See attached waveforms for more clarity):

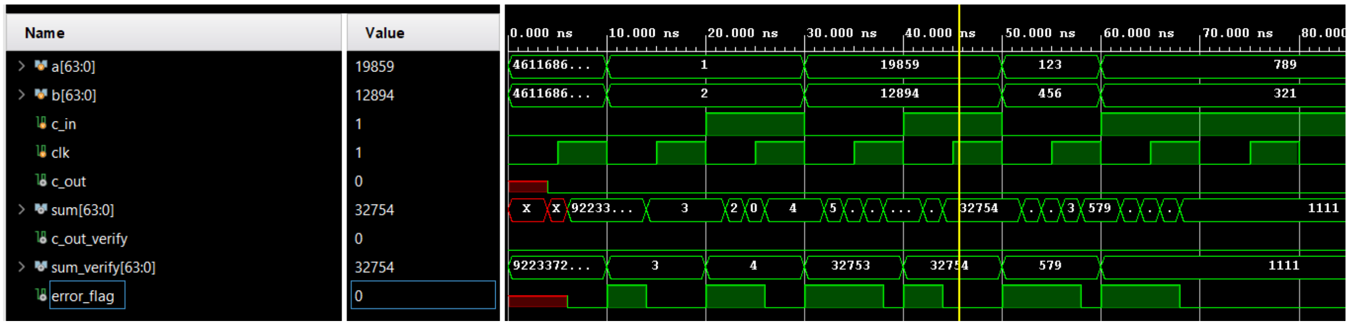


64-bit RCA:



Name	Value
a[63:0]	19859
b[63:0]	12894
c_in	1
clk	0
c_out	0
sum[63:0]	32754
c_out_verify	0
sum_verify[63:0]	32754
error_flag	0

64-bit RCA with 2ns Delay:



Name	Value
a[63:0]	19859
b[63:0]	12894
c_in	1
clk	1
c_out	0
sum[63:0]	32754
c_out_verify	0
sum_verify[63:0]	32754
error_flag	0

Timing Description:
The waveforms with 2ns delay were created as a result of a 1ns delay from the half adder, and another 1ns delay from the 1-bit full adder. As these delays propagate from the half adder up to the 1-bit full adder, eventually all the way up to the 64-bit full adder, you see the sum output changing, until it arrives at its final correct value. However, we can also observe that the width of the error_flag is not constant. This is because the actual delay that is experienced will change depending on the input. For larger inputs (more bits) we can expect to see a longer delay before we arrive at the proper sum, meaning our error_flag will be longer. For smaller inputs, we'll see a lower delay before achieving the correct sum as the bits (and delay) propagate up to our highest level RCA.

Questions:
Since we know a 1-bit full adder has a gate delay of 2, we can scale this up for our 64-bit RCA to get a gate delay of 128 (64 * 2).

If we apply the same logic to the CSA, we only need to add up the gate logic for one of the 32-bit RCAs, since the three that are used in the 2-stage CSA are run in parallel. This means that the entire 2-stage CSA should only have a gate delay of 64 (32 * 2).

These values are not what I would expect based on looking at the waveforms, since they appear to only propagate the original 2ns gate delay from the first half adder and full adder. I suppose the gate delays calculated above are only in worst case scenarios, and it makes sense that the absolute minimum delay we could experience is 2, so maybe my test bench was just sampling inputs that resulted in a lower delay. But I tested a range of inputs including forcing a carry out, and still experienced a lower delay than I would have thought.