Lab 4 EC413: 64-bit Adders

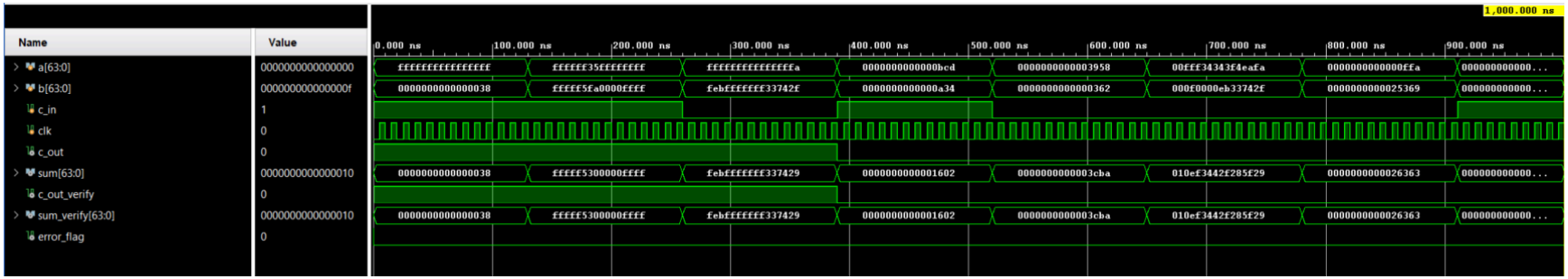
Members: Pree Simphliphan, Rayan Syed

Note: The delay (D) used for the gates in the delay waveforms is always 1 ns in both of the 64 bit adders.

64-bit Ripple Carry Adder [name: Sixty_Four_bit_RCA]:

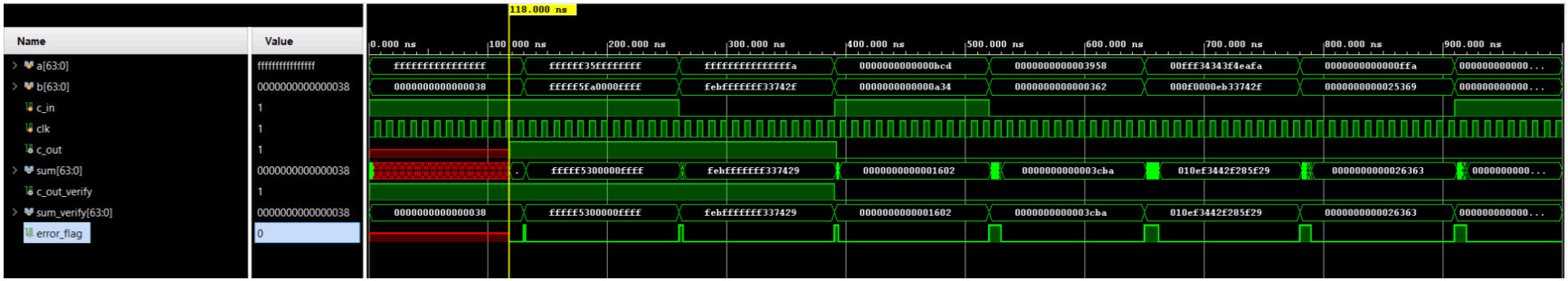
The design: utilize 2 32-bit RCA [name: Thirtytwo_bit_RCA] which is composed of 8 4-bit RCA [name: Four_bit_FA] from preLab. Therefore, this design is hierarchical in which one of the modules is 4-bit ripple carry adder and this design is structural Verilog. We created Verification_64bit by using regular operation to verify in testbench if our design outputs the correct answer or not

Standard Logic Gate Waveform:



Delayed Gate Waveform:

Expected **128 ns worst case** [128 gate delay (2 gate delay per each adder since there are 2 logic gate level and each level, we set the delay equal as 1 ns)], However, this will depend on each test case because in some test cases, the calculation through gates is not required for some digits which means the delay can cause less than 128 ns but 128 ns is the maximum possible delay.



As can be seen in this waveform, the delayed gates cause the 64 bit RCA to have a delay of 118 ns for the first test case (error flag could not even be seen until 118 ns passed). This matches with what we expected since the RCA does not need to calculate all the bits even though this test case is relatively large, so its slightly less than 128 ns. After this, it is evident that the error flag keeps coming on throughout the waveform showing that the delay is in fact present and causing glitches.

Representative Examples:

Force carry out case-> (a = 0xFFFFFFFFFFFFFFFF; b = 56; c_in = 1; the result is sum = 0x38 c_out = 1)

Random Large A&B case with carry-> (a = 0xFFFFFF35FFFFFFFF; b = 0xFFFFF5FA0000FFFF; c_in = 1; the result is sum = 0xFFFFF5300000FFFF c_out = 1)

Random Large A&B case without carry-> (a = 0xFFFFFFFFFFFFFFFA; b = 0xFEBFFFFFFF33742F; c_in = 0; the result is sum = 0xFEBFFFFFFF337429 c_out = 1)

Random Small A&B case with carry-> (a = 0xBCD; b = 0xA34; c_in = 1; the result is sum = 0x1602 c_out = 0)

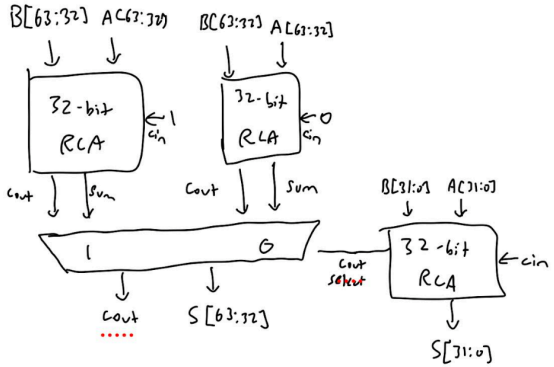
Random Small A&B case with carry-> (a = 0x3958; b = 0x362; c_in = 0; the result is sum = 0x3CBA c_out = 0)

Random Case-> (a = 0xFFF34343F4EAFA; b = 0xF0000EB33742F; c_in = 0; the result is sum = 0x10EF3442F285F29 c_out = 0)

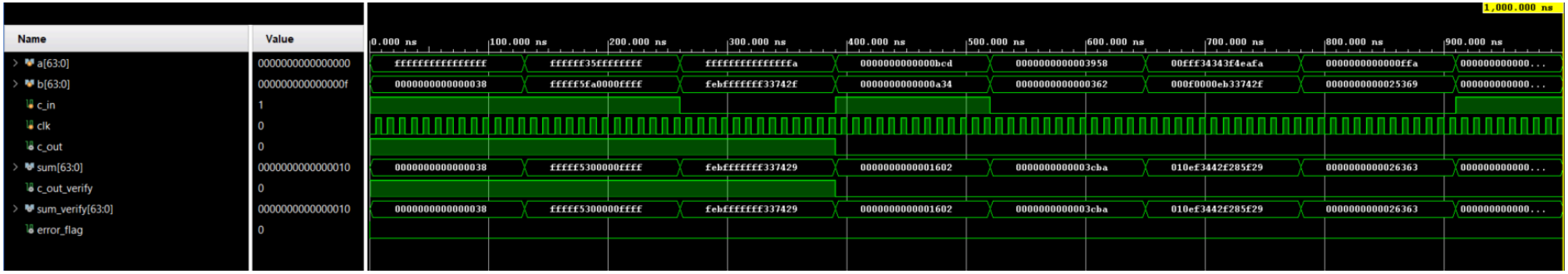
Random Case-> (a = 0xFFA; b = 0x25369; c_in = 0; the result is sum = 0x26363 c_out = 0)

64 bit Carry Select Adder:

The design: Our 2 stage 64 bit Carry Select Adder uses three 32-bit RCAs. The RCA modules used are the same as for the 64 bit RCA. This makes the design hierarchical. The first RCA computes the sum and c_out for the first 32 bits. The sum is saved as part of the output and the c_out is used as a select bit for the mux of the other two 32 bit RCAs. The other two RCAs calculate the sum of the second 32 bits but one assumes the carryin will be 1 and the other 0. This diagram sums up the design we made:

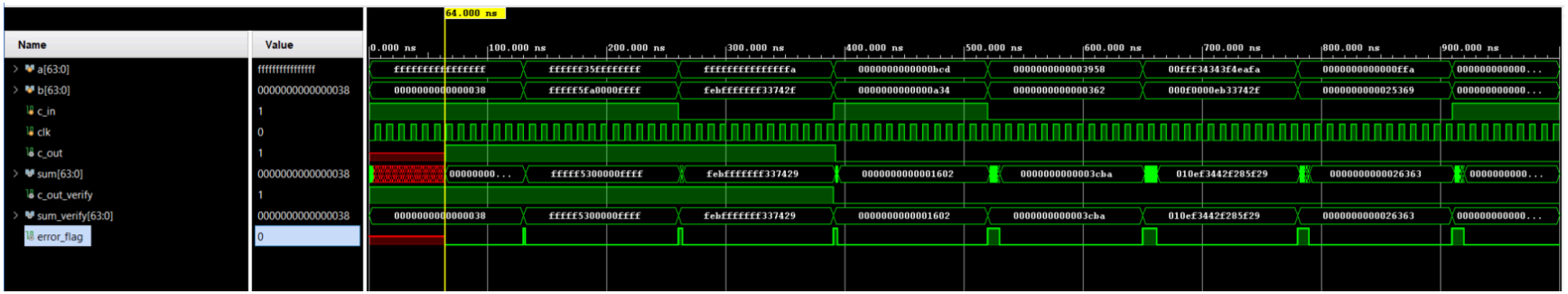


Standard Logic Gate Waveform:



Delayed Gate Waveform:

The expected delay here was **64 ns worst case** (half of the RCA predicted since only the delay for a 32 bit RCA which has 32 FA which should each have 2 ns delay so 32*2 = 64; since it's 2 stages we don't have to take an account of AND-OR gate to connect each stage [2*(#stage - 2)]). The sum could also be calculated in less than 32 ns alternatively if the bitcount is less than 32, since the result of the big select mux will no longer be required in this case.



As can be seen in this waveform, the delay was indeed 64 ns since it took 64 ns for the sum to be outputted (error flag could not even be seen until 64 ns passed). After this, it is evident that the error flag keeps coming on throughout the waveform showing that the delay is in fact present and causing glitches.

The representative examples are the same as for the RCA example since the same values are used in the testbenches. Please refer to that section for the representative example details.