Ande Chen U93195418

Modules

AND_T.v

AND gate with no delay

OR_T.v

OR gate with no delay

AND_D.v

AND gate with delay of 1

OR_D.v

OR gate with delay of 1

FA_str.v

1-bit full adder that is built from the AND and OR gates with no delay

FA_D_str.v

1-bit full adder that is built from the AND and OR gates with delay

FBFA_str.v

This module is a 4-bit full adder that is built from 4 1-bit adders without delay.

FBFA_D_str.v

This module is a 4-bit full adder that is built from 4 1-bit adders with delay.

8BRCA.v

8-bit ripple carry adder that is built from 2 4-bit adders. It takes 2 8-bit inputs a and b and a 1-bit input c_in and produces an 8-bit output sum and a 1-bit output c_out. The first 4-bit adder computes the sum of the least significant 4 bits of the inputs with the carry in. The second 4-bit adder takes the carry out of the first 4-bit adder as its carry in, computes the sum of the most significant 4 bits of the inputs, and produces an overall carry out.

8BRCA_D_str.v

8-bit ripple carry adder with delays

Ande Chen U93195418

16BRCA.v

16-bit ripple carry adder that is built from 2 8-bit adders. It takes 2 16-bit inputs a and b and a 1-bit input c_in and produces a 16-bit output sum and a 1-bit output c_out. The first 8-bit adder computes the sum of the least significant 8 bits of the inputs with the carry in. The second 8-bit adder takes the carry out of the first 8-bit adder as its carry in, computes the sum of the most significant 8 bits of the inputs, and produces an overall carry out.

16BRCA_D_str.v

16-bit ripple carry adder with delays

32BRCA.v

32-bit ripple carry adder that is built from 2 16-bit adders. It takes 2 32-bit inputs a and b and a 1-bit input c_in and produces a 32-bit output sum, and a 1-bit output c_out. The first 16-bit adder computes the sum of the least significant 16 bits of the inputs with the carry in. The second 16-bit adder takes the carry out of the first 16-bit adder as its carry in, computes the sum of the most significant 16 bits of the inputs, and produces an overall carry out.

32BRCA_D_str.v

32-bit ripple carry adders with delay

64BRCA.v

64-bit ripple carry adder that is built from 2 32-bit adders. It takes 2 64-bit inputs a and b and a 1-bit input c_in and produces a 64-bit output sum and a 1-bit output c_out. The first 32-bit adder computes the sum of the least significant 32 bits of the inputs with the carry in. The second 32-bit adder takes the carry out of the first 32-bit adder as its carry in, computes the sum of the most significant 32 bits of the inputs, and produces an overall carry out.

64BRCA_D_str.v

64-bit ripple carry adders with delay

mux.v

This module uses behavioral Verilog to build a 2:1 mux. It takes 2 32-bit inputs in_0 and in_1, a 1-bit input s for select, and a 32-bit output out. If the select is 1, the output is in_1, otherwise, the output is in_0.

2StageCarrySelect.v

This module is a two stage 64-bit carry select adder that is built from 3 32-bit adders and a 2:1 mux. It takes 2 64-bit inputs a and b, a 1-bit input c_in, an 8-bit output sum, and a 1-bit output c_out.

2StageCarrySelect_D_str.v

Two stage 64-bit carry select adder with delay

Verification_64bit.v

This module uses behavioral Verilog to produce a carry out and a sum by computing the sum of 2 64-bit inputs a and b and a 1-bit carry in.

SixtyFourBitRCA_str_tb.v

This module is a testbench to verify that the 64-bit ripple carry adder works as intended by comparing the output of the 64-bit ripple carry adder using structural Verilog to the output from the verification module. It tests cases with an overall carry out, no carry out, large inputs with carry in, large inputs without carry in, small inputs with carry in, small inputs with carry in, and random combinations.

SixtyFourBitRCA_D_str_tb.v

Testbench to verify outputs of the 64-bit ripple carry adder with delay

64CarrySelectAdder_str_tb.v

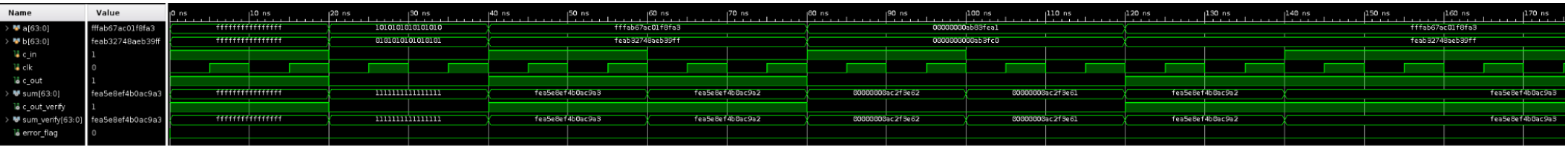
This module is a testbench to verify that the 64-bit carry select adder works as intended by comparing the output of the 64-bit carry select adder using structural Verilog to the output from the verification module. It tests cases with an overall carry out, no carry out, large inputs with carry in, large inputs without carry in, small inputs with carry in, small inputs with carry in, and random combinations.

64CarrySelectAdder_D_str_tb.v

Testbench to verify outputs of the 64-bit carry select adder with delay

Waveform Screenshots (last test case ends at 160 ns, 8 tests and 20ns wait for each)

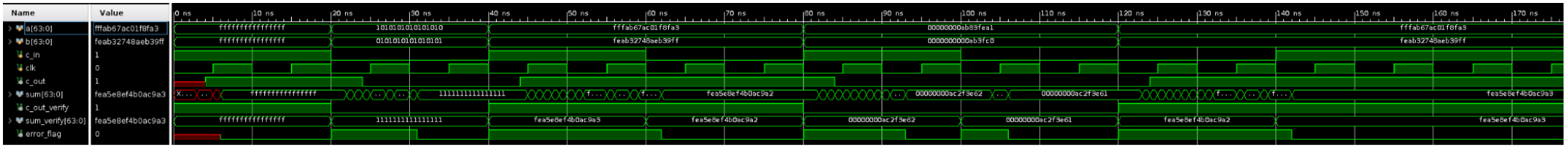
64-bit Ripple Carry Adder (No Delay)



There is no delay, so the sum is computed instantaneously.

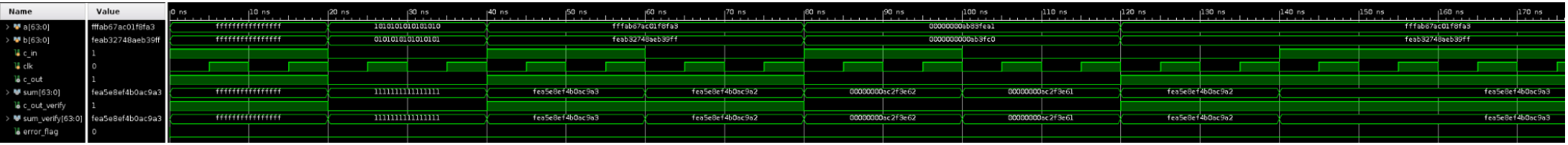
Ande Chen U93195418

64-bit Ripple Carry Adder (With Delay)



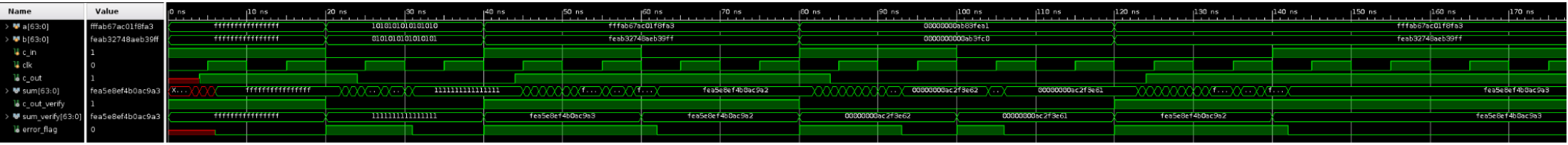
The delays from the AND and OR gates propagate throughout the adder, so the final answer does not compute until after the delay. The duration of the delay is different for different inputs because the inputs are not always computed with the longest path through the adder.

64-bit Carry Select Adder (No Delay)



There is no delay, so the sum is computed instantaneously.

64-bit Carry Select Adder (With Delay)



The delays from the AND and OR gates propagate throughout the adder, so the final answer does not compute until after the delay. The duration of the delay is different for different inputs because the inputs are not always computed with the longest path through the adder.

Ande Chen U93195418

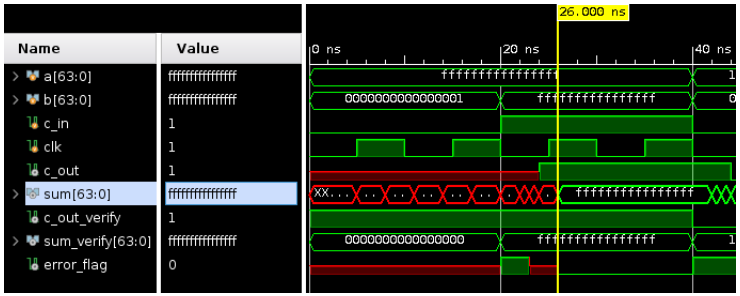
Questions

What are the times of your adders in gate delays? Are they what you expect? Why or why not?

Ripple Carry w/o Delay = 0 gate delay

This is expected, because if the AND and OR gates have no delay, then building the adders from those gates doesn't introduce further delays. The sum always computes instantaneously.

Ripple Carry w/ Delay = 26000 Gate Delay

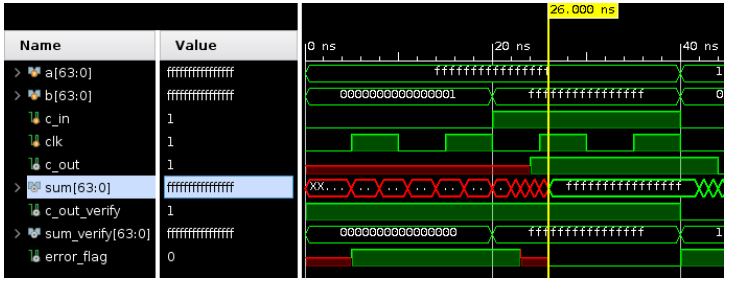


This is expected because the delay of 1 needs to propagate from the simple AND and OR gates all the way up through the 64-bit ripple carry adder.

Carry Select w/o Delay = 0 gate delay

This is expected, because if the AND and OR gates have no delay, then building the adders from those gates doesn't introduce further delays. The sum always computes instantaneously.

Carry Select w/ Delay = 26000 Gate Delays



This is not expected. There should still be a gate delay because each of the AND and OR gates have a delay of 1. However, the overall gate delay for the carry select adder should be half of that of the ripple carry adder, because the sum of the higher order 32-bits is calculated in parallel with the lower order 32-bits. This should cut the computation time down by half.