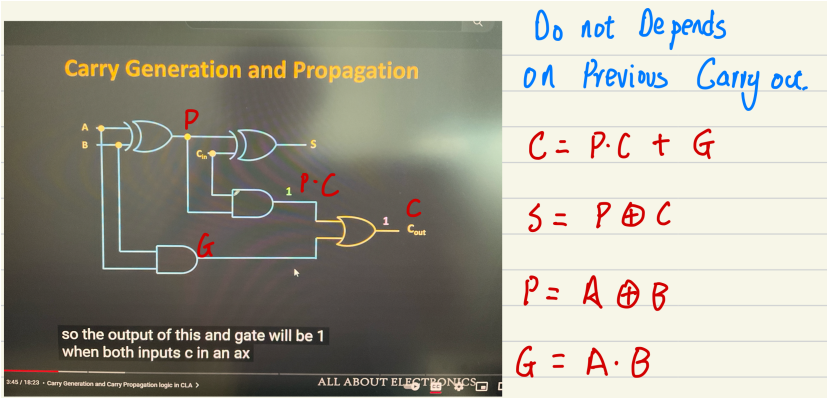
ECE260A Lab3

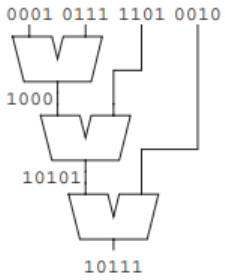
Chengming Li

A59026442

Ee260afa23cq

1. Design Choice
    a. During this lab, I have tried three different architectures, which are provided by Prof.Eldon. They are Carry save adder(csa_with_cla), the Normal Ripple Carry Adder(fir4rca_u), and Ripple Carry Adder with carry save adder(fir4rca_cas_u).
    b. My final proposed architecture is Carry Save Adder due to its lowest delay, power consumption and minimum area used
    c. Carry Save Adder(csa_with_cla) has the best performance compared to all the other 3. There are few reasons listed below:
        i. Less comb logic in the circuit, which means less propagation delay to get the final result, sum
        ii. Lots of bit wise manipulation, this also save some speed from doing ADD
        iii. Carry out bit does not depend on previous Carry out, so the improvement of delay is better than the other architecture
        iv. Parallelism, carry save adder perform addition in parallel stage.Again it alleviate the dependency to the previous stages
        v. 

This YouTube Video inherently helps me to understand the CSA.
    d. Normal Ripple Carry Adder(fir4rca_u)



(a)

i.

ii. Carryout and Sum dependency: as seen in the figure, the second and third stages depend on the previous stage's output.

iii. Complexity: In the 3rd, the addition is performed between 5 bit and 4 bit input, this add additional complexity into the comb logic

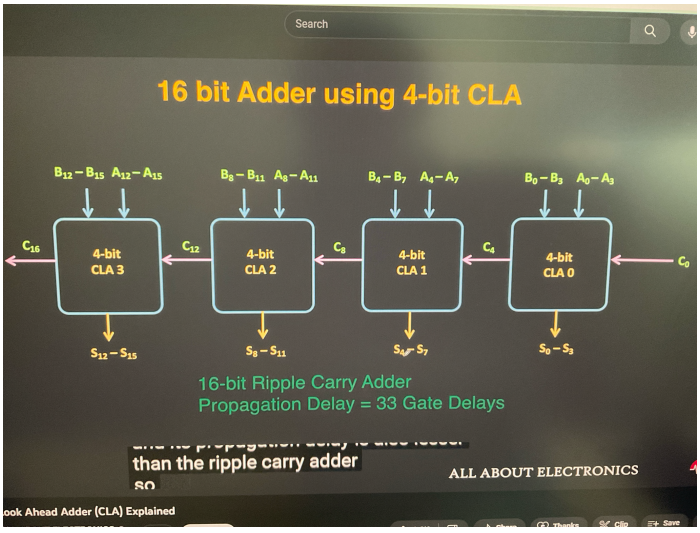
e. Ripple Carry Adder with carry save adder(fir4rca_cas_u)

i. Codewise, there is more comb logic in the circuit.

ii. More dependencies compared to the previous two, less parallelism

2. Future Iteration

a. There is another architecture I found online, which makes existing CSA more parallel.

b. 

c. Only 4 bit CLA is needed to be created and cascade them together.

d. Based on the ideal assumption, i.e., every block has equal delay, this architecture can be 4 times faster than the ripple carry adder.

3. Table of Power, Performance and area comparison

Metrics	Ripple Carry Adder	Carry Saver Adder	fir4rca_u	fir4rca_cas_u
Worst negative slack in ps	-0.178	-0.222	-0.185	-0.366
Total power(w)	1.817mw	1.66mW	1.8226mW	1.93mW
Leakage power	25.2uW	18.04uW	25.91uW	29.15uW
Area of comb logic	1820.52	1142.64	1891.44	2242.44
# of comb cells	826	524	860	961
Total area	2457.36	1771.2	2528.28	2846.52

4. Conclusion:
    a. Carry Saver Adder has a clear win over the other three architectures due to parallelism and less dependencies of Carry out in the delay manner.
    b. However, the power consumption of CSA has a small win over the other three architectures. Because CSA reduces the cascading nature of the adder.
    c. In the real design, if there are more bits addition, the cost and the complexity of the logic design increase. And the power dissipation will also increase due to the complexity of logic.
    d. For future iterations, the cascade 4 bits CSA is introduced in the last page. This could save more speed, but at the cost of Area of comb logic and power dissipation.
5. Pointer to my proposed SV files
    a. Carry Save Adder(csa_with_cla)
        i. /home/linux/ieng6/ee260afa23/ee260afa23cq/lab3/dc_proposed_design/rtl
        ii. /home/linux/ieng6/ee260afa23/ee260afa23cq/lab3/dc_proposed_design/log
    b. fir4rca_u
        i. /home/linux/ieng6/ee260afa23/ee260afa23cq/lab3/fir4rca_u/rtl
        ii. /home/linux/ieng6/ee260afa23/ee260afa23cq/lab3/fir4rca_u/log
    c. Fir4rca_cas_u
        i. /home/linux/ieng6/ee260afa23/ee260afa23cq/lab3/fir4rca_cas_u/rtl
        ii. /home/linux/ieng6/ee260afa23/ee260afa23cq/lab3/fir4rca_cas_u/log