컴퓨터 공학 기초 실험2 보고서

실험제목: Carry Look-ahead Adder(CLA)

실험일자: 2023년 09월 22일 (금)

제출일자: 2023년 10월 01일 (일)

학 과: 컴퓨터공학과

담당교수: 이형근 교수님

실습분반: 금요일 0, 1, 2

학 번: 2020202037

성 명: 염정호

1. 제목 및 목적

A. 제목

Carry Look-ahead Adder(CLA)

B. 목적

4bit carry look ahead를 구현하고 이를 이용해 32bit CLA모듈을 구현한다.

구현한 32bit-cla에 앞뒤로 flip-flop를 추가하여 유효한 결과값이 나오는데 걸리는 시간을 측정한다.

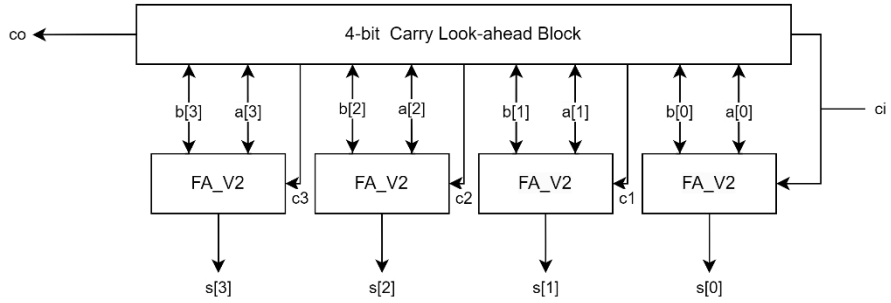
이전에 만든 4bit-rca를 직렬로 연결한 32bit-rca 모듈과 소요 시간을 비교해 본다.

2. 원리(배경지식)

CLA : RCA가 계산이 완료될 때 까지의 시간이 오래 걸리는 단점을 보완한 회로로 입력 a, b, carry가 주어질 때, 모든 자리 올림수가 동시에 구해져 계산시간을 단축 할 수 있도록 설계한 가산기 이다. carry만을 계산해주는 carry_look-ahead block이 존재한다.

Timing Analysis(시간분석) : 디지털 시스템 설계 및 최적화에서 중요한 역할을 하는 프로세스이다. 주로 고성능 디지털 시스템에서 클럭, 동기화 및 지연과 같은 시간 요소들을 분석/최적화 하는데 이용된다. Timing Analysis는 회로의 동작을 예측하고, 디지털 시스템이 시간에 따라 어떻게 동작할지 이해하는데 도움을 준다.

3. 설계 세부사항



CLA_Block구조

a	b	Cin	Cout	Condition
0	0	0	0	No carry Generation
0	0	1	0	
0	1	0	0	
0	1	1	1	No carry

1	0	0	0	Propagate
1	0	1	1	
1	1	0	1	Carry generate
1	1	1	1	

CLA truth table

Carry Look-ahead Block Boolean Equation

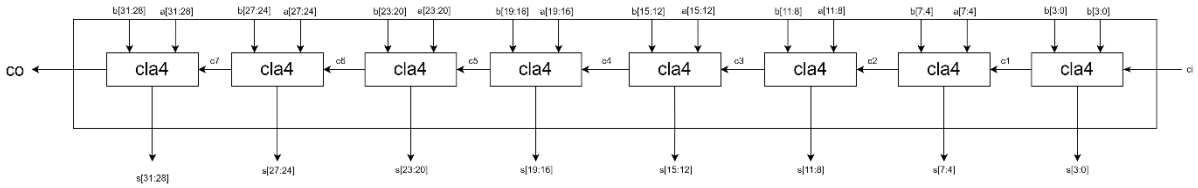
$$G_i = A_i B_i \qquad P_i = A_i + B_i$$

Full adder의 carry out에 적용하면 :

$$C_{i+1} = A_i B_i + (A_i + B_i) C_i = G_i + p_i C_i$$

이를 이용해 4bit CLA의 carry를 계산해 보면

$$C_1 = G_0 + P_0 ci$$
$$C_2 = G_1 + P_1 G_0 + P_1 P_0 C_0$$
$$C_3 = G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_0 G_0 + P_2 P_1 P_0 C_0$$
$$C_{cout} = G_3 + P_3 G_2 + P_3 P_2 P_1 G_1 + P_3 P_2 P_1 G_0 + P_3 P_2 P_1 P_0 C_0$$



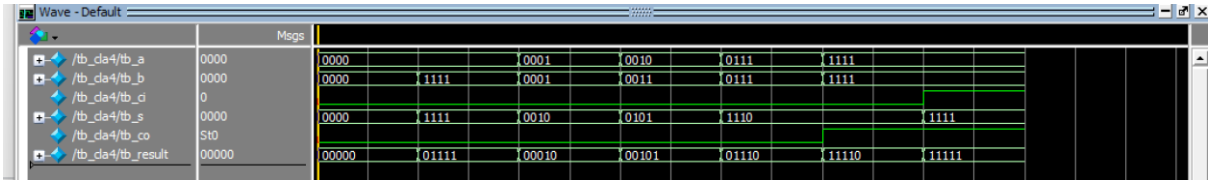
32-bit Cla 구성

4 bit cla모듈을 직렬로 연결하여 구할 수 있다. i번째 cla의 Cout을 i+1번째 Cla의 Cin과 연결된다.

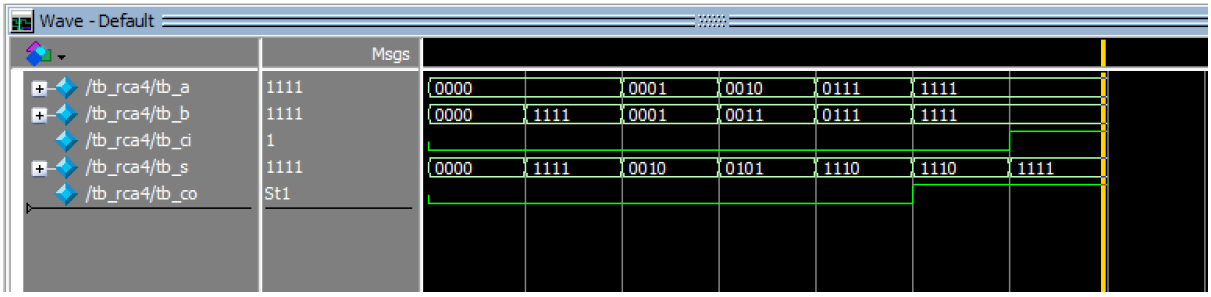
4. 설계 검증 및 실험 결과

A. 시뮬레이션 결과

설계한 디자인을 검증하기 위하여 작성한 testbench에 대하여 간단하게 설명하고, waveform을 통하여 원하는 결과가 제대로 나오는 지를 확인한다.



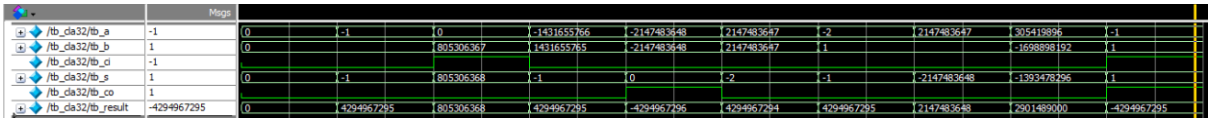
4bit cla



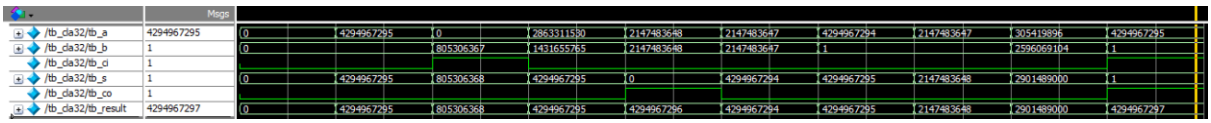
4bit rca

두 테스트벤치의 결과 값이 같은 것을 확인 할 수 있다. cla모듈이 정상 작동함을 나타냄

32bit -cla



signed



Unsigned

```
    begin
    tb_a = 32'h0; tb_b = 32'h0; tb_ci = 0;
    #10; tb_a = 32'hffff_ffff; tb_b = 32'h0000_0000; // No carry, all sum = 15
    #10; tb_a = 32'h0000_0000; tb_b = 32'h2fff_ffff; tb_ci = 1; // All carry, all sum = 0
    #10; tb_a = 32'haaaa_aaaa; tb_b = 32'h5555_5555; tb_ci = 0;// A: 1010... / B: 0101...
    #10; tb_a = 32'h8000_0000; tb_b = 32'h8000_0000; // Overflow test signed
    #10; tb_a = 32'h7fff_ffff; tb_b = 32'h7fff_ffff; // No overflow test signed
    #10; tb_a = 32'hffff_fffe; tb_b = 32'h0000_0001; // Overflow test unsigned
    #10; tb_a = 32'h7fff_ffff; tb_b = 32'h0000_0001; // No overflow test unsigned
    #10; tb_a = 32'h1234_5678; tb_b = 32'h9abc_def0; // Random values
    #10; tb_a = 32'hffffffff; tb_b = 32'h0000_0001; tb_ci = 1; // All ci test & overflow
    #10; $stop;
```

주석에서 알 수 있드시 모든 sum이 발생한 경우의 값이 제대로 출력되는지,

overflow없이 모든 carry가 발생했을 때 출력값

usigned / signed이 때 각각의 over flow발생했을 때와 발생 하지 않았을 때 출력을 테스트 하였고

마지막으로 랜덤한 값을 넣어 테스트 진행하였다.

Slow 1100mV 85C Model

	Fmax	Restricted Fmax	Clock Name	Note
1	145.73 MHz	145.73 MHz	clk	

Cla : 145.73MHz

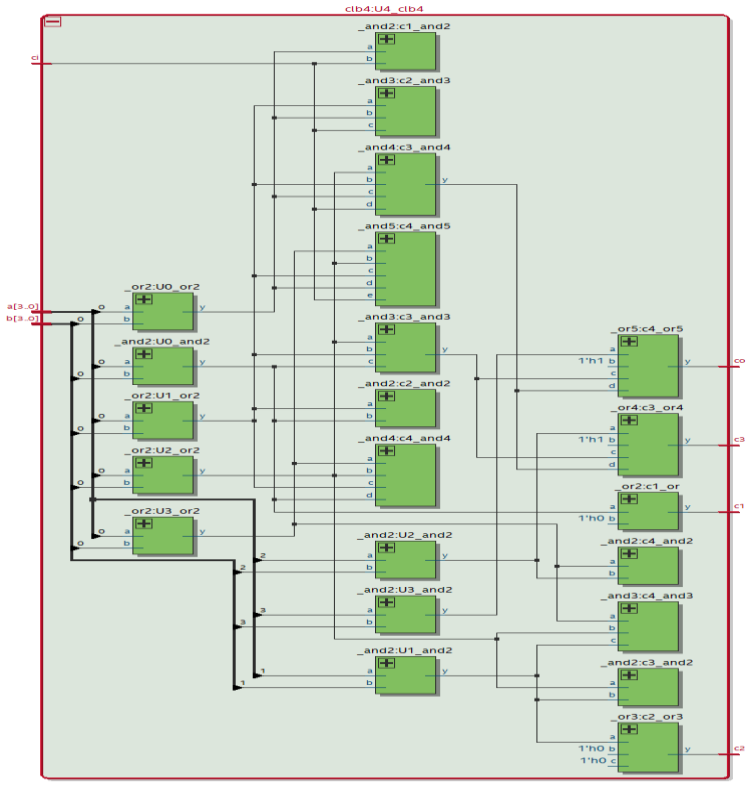
<<Filter>>

	Fmax	Restricted Fmax	Clock Name	Note
1	164.31 MHz	164.31 MHz	clk	

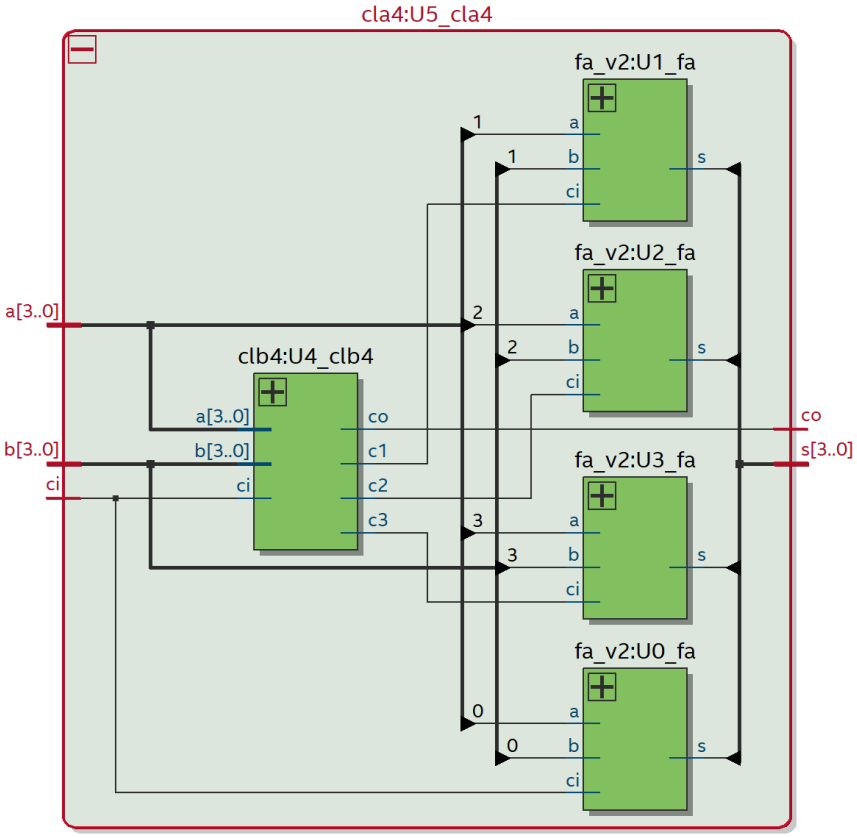
Rca: 164.31MHz

시뮬레이션 결과 Cla의 연산속도가 Rca보다 빨랐다.

B. 합성(synthesis) 결과

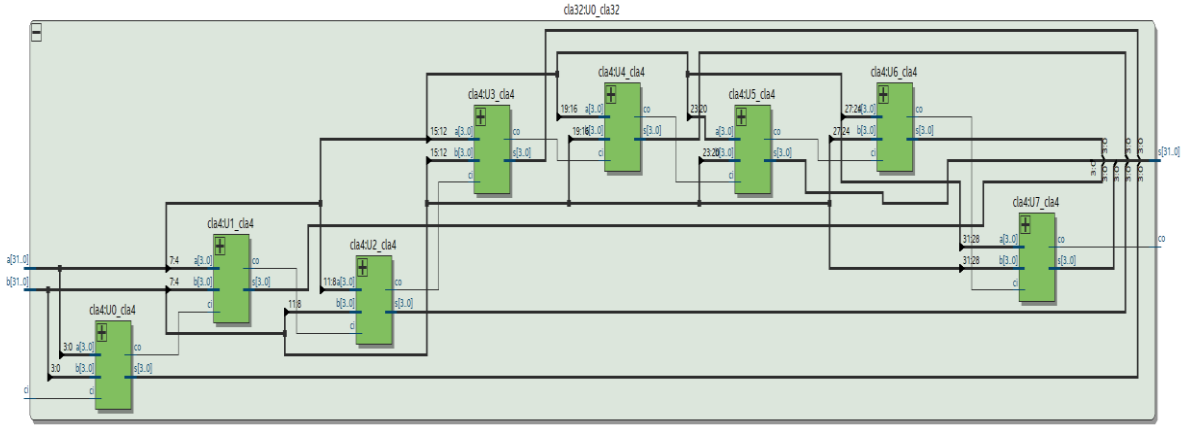


cla_block(Gi , Pi를 계산)

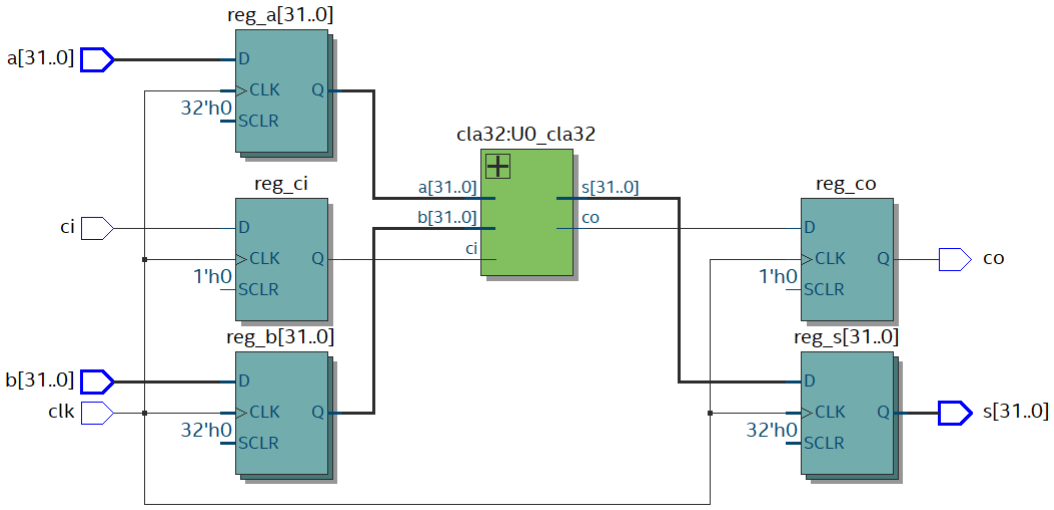


4bit-Cla

Cla에서는 fulladder의 carry out은 필요하지 않다.



32bit-cla(4bit-cla 8개가 직렬로 연결)



Cla_clk

32bit-cla에 앞뒤로 filp-flop를 추가해 clock과 연결한 모듈이다.

Clock을 이용해 결과값을 확인하기 위해 사용된다.

Flow Summary

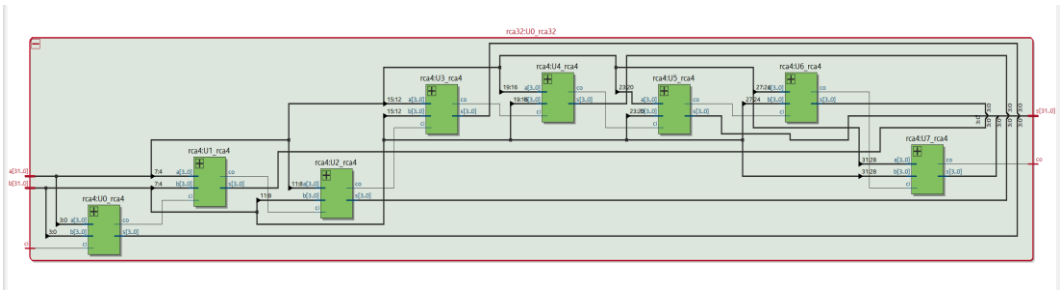
<<Filter>>

Flow Status	Successful - Sun Oct 01 12:05:49 2023
Quartus Prime Version	18.1.0 Build 625 09/12/2018 SJ Lite Edition
Revision Name	cla_clk
Top-level Entity Name	cla_clk
Family	Cyclone V
Device	5CSXFC6D6F31C6
Timing Models	Final
Logic utilization (in ALMs)	33 / 41,910 (< 1 %)
Total registers	98
Total pins	99 / 499 (20 %)
Total virtual pins	0
Total block memory bits	0 / 5,662,720 (0 %)
Total DSP Blocks	0 / 112 (0 %)
Total HSSI RX PCSs	0 / 9 (0 %)
Total HSSI PMA RX Deserializers	0 / 9 (0 %)
Total HSSI TX PCSs	0 / 9 (0 %)
Total HSSI PMA TX Serializers	0 / 9 (0 %)
Total PLLs	0 / 15 (0 %)
Total DLLs	0 / 4 (0 %)

32bit – cla Flow Summary

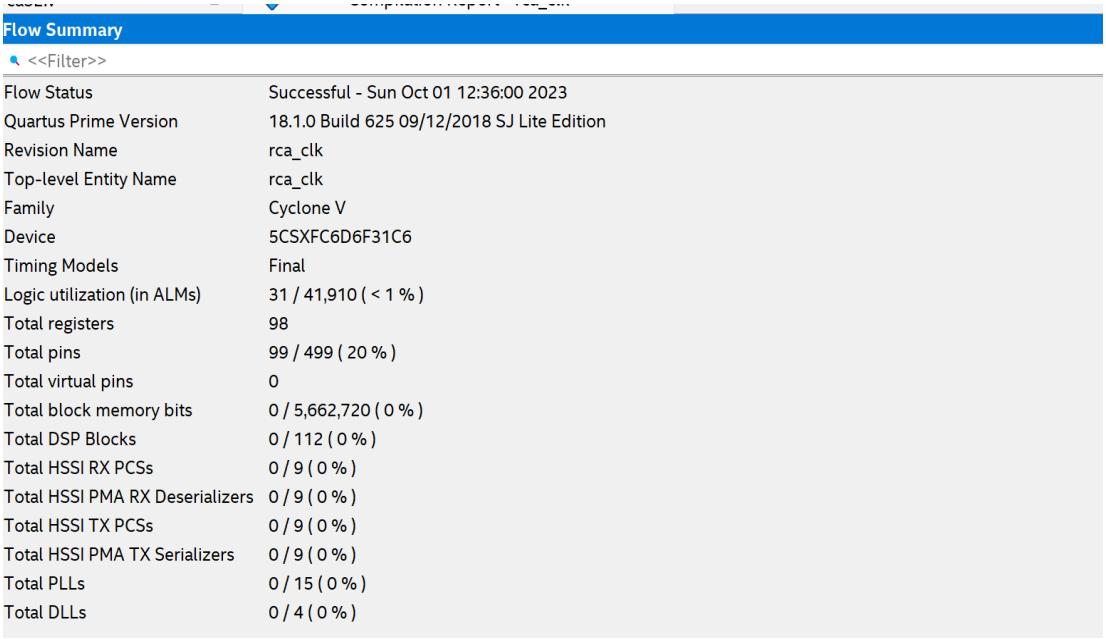
98개의 reigister 사용

99개의 핀 input a(32), b(32), ci(1), clk(1), s(32), co(1)이 사용되었다.



32bit-rca

4bit-rca 8개가 직렬로 연결된 형태이다.



Flow Summary

<<Filter>>

Flow Status	Successful - Sun Oct 01 12:36:00 2023
Quartus Prime Version	18.1.0 Build 625 09/12/2018 SJ Lite Edition
Revision Name	rca_clk
Top-level Entity Name	rca_clk
Family	Cyclone V
Device	5CSXFC6D6F31C6
Timing Models	Final
Logic utilization (in ALMs)	31 / 41,910 (< 1 %)
Total registers	98
Total pins	99 / 499 (20 %)
Total virtual pins	0
Total block memory bits	0 / 5,662,720 (0 %)
Total DSP Blocks	0 / 112 (0 %)
Total HSSI RX PCSs	0 / 9 (0 %)
Total HSSI PMA RX Deserializers	0 / 9 (0 %)
Total HSSI TX PCSs	0 / 9 (0 %)
Total HSSI PMA TX Serializers	0 / 9 (0 %)
Total PLLs	0 / 15 (0 %)
Total DLLs	0 / 4 (0 %)

32bit-rca Flow summary

98개의 reigister 사용

99개의 핀 input a(32), b(32), ci(1), clk(1), s(32), co(1)이 사용되었다.

Cla와 Rca에서 사용된 레지스터와 핀의 개수는 같다.

5. 고찰 및 결론

A. 고찰

Quartus프로그램에서 오류코드 잡는 것이 어려웠다. 보고서 작성을 위한 test bench작성 중 코드에 상당히 많은 오류가 있음을 확인했다. 선언하지 않은 변수임에도 따로 오류가 발생하지 않고 코드 실행이 가능해 코드 문법에 문제가 있는게 아니라면 에러코드가 발생하지 않음을 느꼈다.

테스트벤치 작성시 고려할 사항이 많다는 것을 느꼈다. 첫 실험에서 tb작성 시 모든 sum이 발생했을 때, 모든 carry가 발생했을 때 만 고려했는데, 이번 과제를 진행하면서 각각 변수를 signed, unsigned로 했을 때, 오버플로가 발생했을 때 등 고려할 사항들이 많다는 것을 느꼈다.

B. 결론

해당 실험에서 RCA와 CLA의 계산 속도 차이를 비교해 보았다. 둘의 연산 결과값이 같았

고, 레지스터의 개수, 사용되는 input, output핀의 개수 또한 같았다. 하지만 연산속도는 Cla 145MHz / Rca 165MHz로 Cla가 더 빨랐다. rca는 모든 carry가 순차적으로 발생하는 반면에 32bit - cla에서는 8개의 4bit-cla가 동시에 연산되기 때문에 cla의 연산 속도가 rca에 비해 빠르다는 것을 실험으로 확인했다.

또한 우리가 실험으로 구현한 32bit cla와 xor게이트를 사용하지 않고 or, and게이트를 이용한 modified 32-bit CLA의 경우 사이즈도 더 작았고 속도도 10%정도 향상된 것을 확인할 수 있었다.

6. 참고문헌

CLA /https://ko.wikipedia.org/wiki/%EC%9E%90%EB%A6%AC%EC%98%AC%EB%A6%BC%EC%88%98_%EC%98%88%EC%B8%A1_%EA%B0%80%EC%82%B0%EA%B8%B0

timing analysis / https://en.wikipedia.org/wiki/Static_timing_analysis

이준환/디지털논리회로/광운대학교/2023

이형근/컴퓨터공학기초실험2/광운대학교/2023