컴퓨터 공학 기초 실험2 보고서

실험제목: Carry Look-ahead Adder (CLA)

실험일자: 2021년 09월 28일 (수)

제출일자: 2021년 10월 04일 (화)

학 과:컴퓨터정보공학부

담당교수: 공진흥 교수님

실습분반: 수요일 0, 1, 2

학 번: 2019202021

성 명: 정 성 엽

1. 제목 및 목적

A. 제목

Carry Look-ahead Adder (CLA)

B. 목적

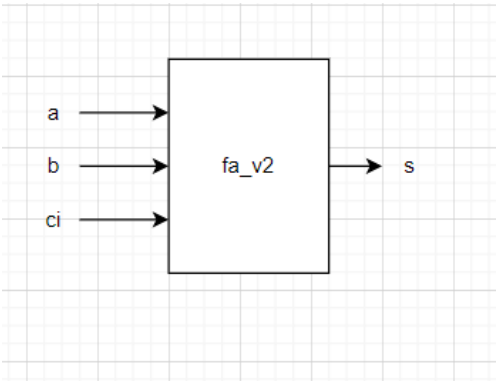
Full Adder에서 carry out 값만 미리 계산할 수 있도록 하는 clb4를 설계하고 이를 이용해 cla4 모듈을 설계한다. 또한 cla4를 이용하여 32 bit의 cla를 설계하고 Flip Flop을 사용하여 Timing Analysis를 수행한다. 또한 저번 실험에서 사용한 rca4를 이용하여 32 bit의 rca를 설계하고 Timing Analysis 수행한다. 그 후 이 둘을 비교한다.

2. 원리(배경지식)

A. Full Adder Ver.2

기존의 Full Adder는 carry out과 sum을 모두 output으로 계산하였으나 이번 실험에서 사용하는 Full Adder는 input a, b, ci를 입력 받고 가산하여 s만 출력하도록 한다.

input			output
a	b	ci	s
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1



카르노 맵을 그릴 필요도 없이 위 진리표를 보면 1이 홀수의 개수일 때 output s가 1이 됨을 볼 수 있다. 즉 fa_v2의 논리식은 $a \oplus b \oplus ci = s$라고 할 수 있다.

B. Carry look ahead block (clb) & Carry look ahead (cla)

후에 사용할 cla에서는 carry out을 연산하지 않고 clb에서 미리 연산하여 전달한다 이는 carry out이 연속적으로 전달되어 bit 수가 많아지면 속도가 느려지기 때문으로 미리 계산한 carry out 값을 이용해 cla에서 연산한다.

i. 4bit clb

clb에서 carry out을 미리 계산하는 식은 $C_{i+1} = G_i + P_i C_i$ 와 같다. 이때 G_i 는 Generate signal으로 $A_i B_i$와 같고 P_i는 Propagation signal로 $A_i + B_i$와 같다. 이 식을 기반으로 연계하면 4bit인 cla의 carry out을 미리 계산할 수 있다.

$$C_1 = G_0 + P_0 C_{in}$$

$$C_2 = G_1 + P_1 C_1 = G_1 + P_1(G_0 + P_0 C_{in}) = G_1 + P_1 G_0 + P_1 P_0 C_{in}$$

$$C_3 = G_2 + P_2 C_2 = G_2 + P_2(G_1 + P_1 G_0 + P_1 P_0 C_{in}) = G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_{in}$$

$$C_{out} = G_3 + P_3 C_3 = G_3 + P_3(G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_{in})$$
$$= G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0 + P_3 P_2 P_1 P_0 C_{in}$$

위의 식을 이용하여 carry out들을 미리 계산한다.

ii. 4bit cla & 32bit cla

4bit cla는 fa_v2를 이용하여 4bit rca를 구현했을 때와 마찬가지로 나열하여 계산하되 다른점은 각각 full adder가 co로 연결되지 않고 clb에서 미리 계산한 값을 입력 받아 계산한다는 점이다.

그리고 위의 4bit cla를 8개를 직렬로 연결하면 32bit cla를 구현할 수 있다.

C. Timing Analysis

32bit의 cla의 input과 output에 D-Flip Flop을 추가하여 clock의 변화에 따라 값이 업데이트 해주도록 하여 이 clock에 동기화 된 cla를 제작하여 Timing Analysis를 실행할 수 있다. 처음 Timing Analysis를 실행할 경우 음수가 나타날 수 있는데 이는 clock 주기를 조정하여 수정할 수 있다.

D. 32bit rca

이전 실험에서 사용한 rca4 모듈을 위의 32bit cla와 같이 8개를 직렬로 연결하면 32bit의 rca를 구현할 수 있다.

3. 설계 세부사항

A. clb4 module

구분	이름	비트
input	a	4
	b	4
	ci	1
output	c1	1
	c2	1
	c3	1
	co	1

$$C_1 = G_0 + P_0 C_{in}$$

$$C_2 = G_1 + P_1 C_1 = G_1 + P_1(G_0 + P_0 C_{in}) = G_1 + P_1 G_0 + P_1 P_0 C_{in}$$

$$C_3 = G_2 + P_2 C_2 = G_2 + P_2(G_1 + P_1 G_0 + P_1 P_0 C_{in}) = G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_{in}$$

$$C_{out} = G_3 + P_3 C_3 = G_3 + P_3(G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_{in})$$
$$= G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0 + P_3 P_2 P_1 P_0 C_{in}$$

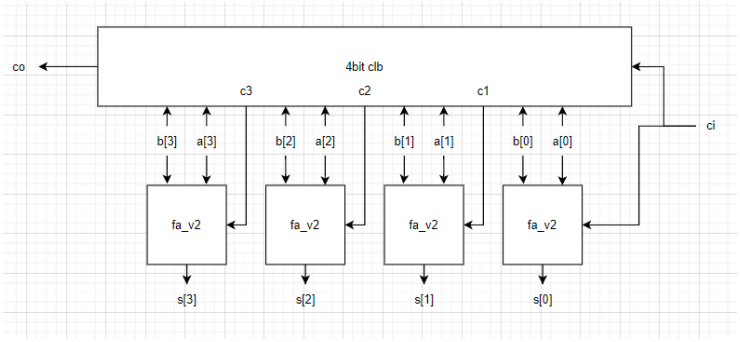
앞서 원리(배경지식)에 설명한 것과 같이 c1~c3 그리고 co에 대해 위의 논리식에 의

거하여 Carry Out을 연산해 output으로 두었다.

B. cla4 module

구분	이름	비트
input	a	4
	b	4
	ci	1
output	co	1
	s	4
wire	c	3

구분	이름
module	cla4
sub module	U0_fa_v2
	U1_fa_v2
	U2_fa_v2
	U3_fa_v2
	U4_clb

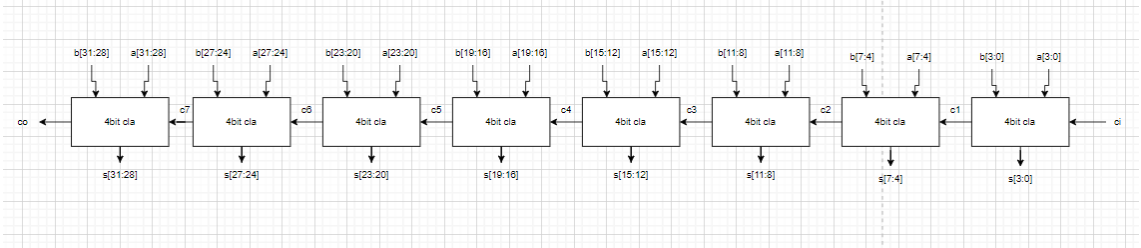


실험에서 제시한 input output wire 그리고 instance name에 따라 verilog를 설계하고 설계 내용은 위의 회로와 같다. Rca4와 같이 carry out을 full adder에서 계산하는 것이 아닌 clb module에서 미리 계산해서 fa 모듈에 전달하는 형태로 설계하였다.

C. cla32 module

구분	이름	비트
input	a	32
	b	32
	ci	1
output	co	1
	s	32
wire	c	8

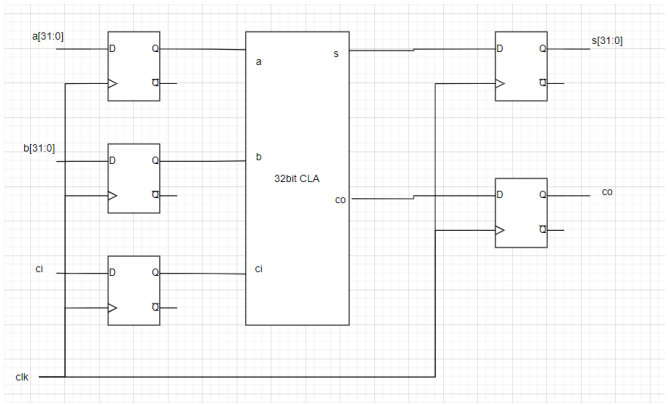
구분	이름
module	cla32
sub module	U0_cla4
	U1_cla4
	U2_cla4
	U3_cla4
	U4_cla4
	U5_cla4
	U6_cla4
	U7_cla4



위 input과 output 그리고 instance를 고려해 설계하였으며 앞서 원리에서 설명한 것과 같이 cla4를 8개를 연결하여 구현하였다.

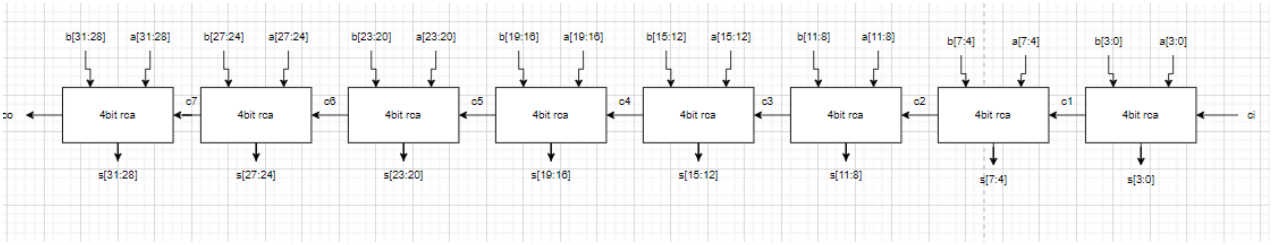
D. cla_clk module

구분	이름	비트
input	clk	1
	a	32
	b	32
	ci	1
output	co	1
	s	32
reg	reg_a	32
	reg_b	32
	reg_ci	1
	reg_s	32
	reg_co	1
wire	wire_s	32
	wire_co	1



Input, output 그리고 reg를 고려하여 verilog를 설계하였으며 wire은 필요에 따라 추가하였다. 위 회로처럼 D-flipflop을 이용하여 32-bit cla를 clock에 동기화 하였다.

E. rca32 module



rca 32 module은 cla 32 module과 같이 rca4를 8개를 직렬로 연결하여 구현하였다.

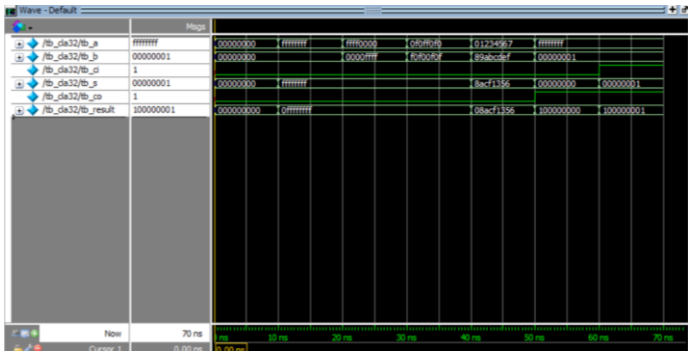
4. 설계 검증 및 실험 결과

A. 시뮬레이션 결과

i. 4bit cla

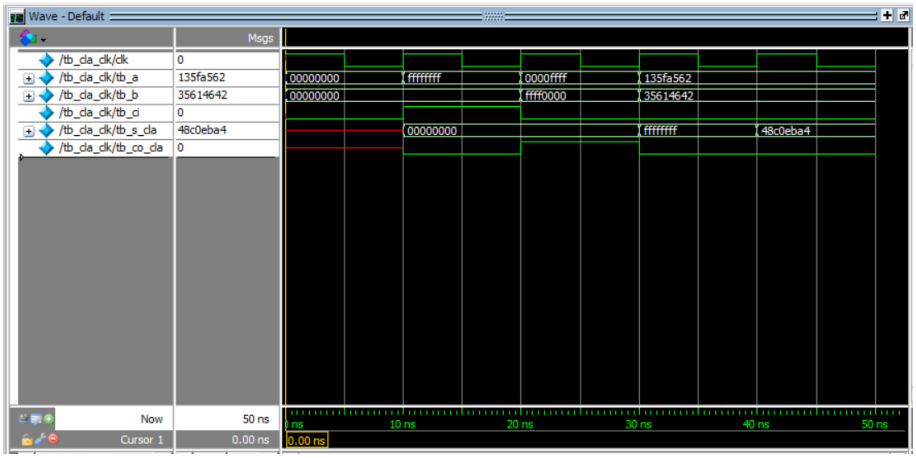
저번 실험에서 구현하였던 4bit rca와 같은 가산을 함을 rca4의 waveform을 통해 확인하였다.

ii. 32bit cla



32bit로 testbench를 돌렸을 때 화면이며 32bit에 맞춰 16진수 8개를 사용하였다. Waveform을 통해 cla를 통해서도 연산이 잘 되었음을 확인할 수 있다.

iii. cla_clk



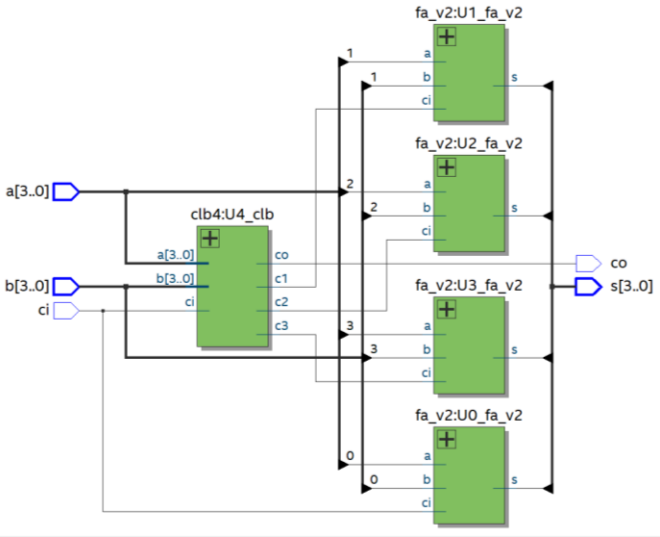
Clock이 rising edge에서 동작하므로 다음 rising edge 타이밍에 나오는 것을 확인할 수 있다. 입력 값에 따라 결과값 s는 a+b의 값 co는 carry out이 출력되는 것을 waveform을 통해 확인할 수 있다.

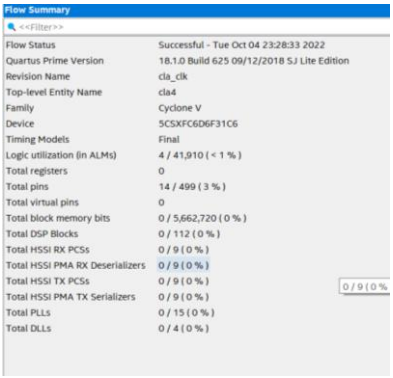
iv. rca_clk

32bit cla_clk와 같은 결과임을 확인할 수 있다.

B. 합성(synthesis) 결과

i. 4bit cla

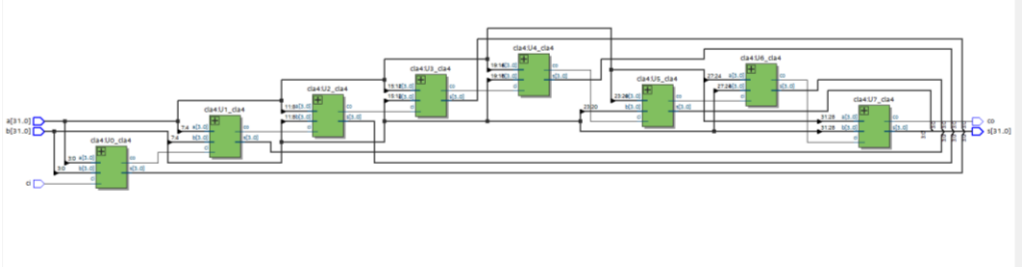


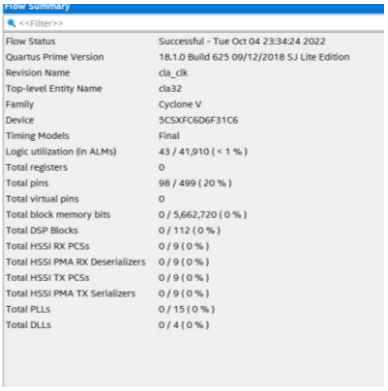


Flow Summary	
Flow Status	Successful - Tue Oct 04 23:28:33 2022
Quartus Prime Version	18.1.0 Build 625 09/12/2018 SJ Lite Edition
Revision Name	cla_clk
Top-level Entity Name	cla4
Family	Cyclone V
Device	5CSXFC6D6F31C6
Timing Models	Final
Logic utilization (in ALMs)	4 / 41,910 (< 1 %)
Total registers	0
Total pins	14 / 499 (3 %)
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 %)

기존 rca4보다 더 복잡한 구조를 가짐을 확인할 수 있다.

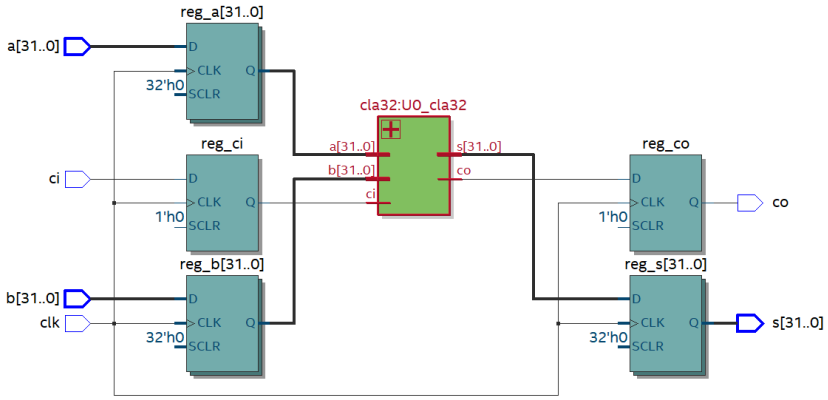
ii. 32bit cla





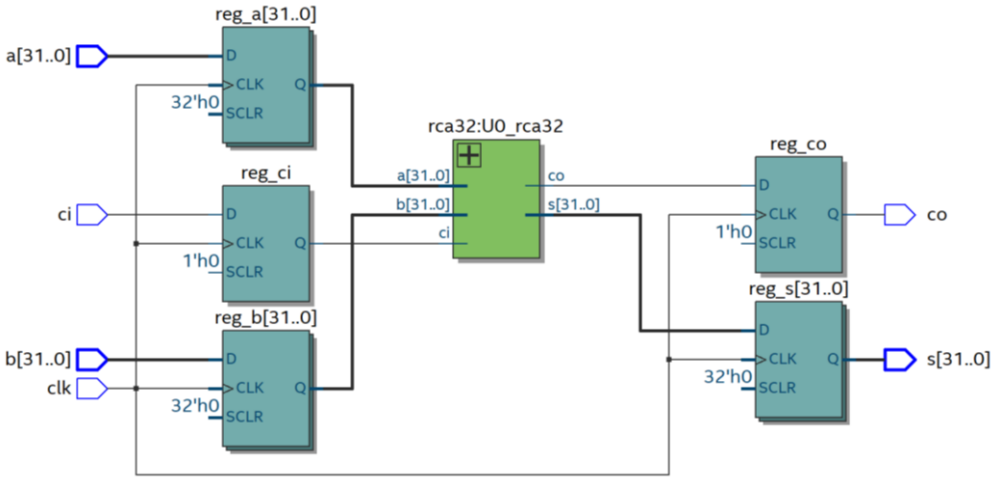
32bit cla 또한 32bit rca에 비해 더 큰 logic utilization임을 확인할 수 있고, 8개의 cla4가 연결됨을 rtl viewer를 통해 확인하였다.

iii. cla_clk



RCA보다 많은 Logic utilization을 볼 수 있는데 이는 32bit cla가 32bit rca보다 더 큰 크기의 회로이며 더 많은 게이트를 사용하는 것을 말한다.

iv. rca_clk



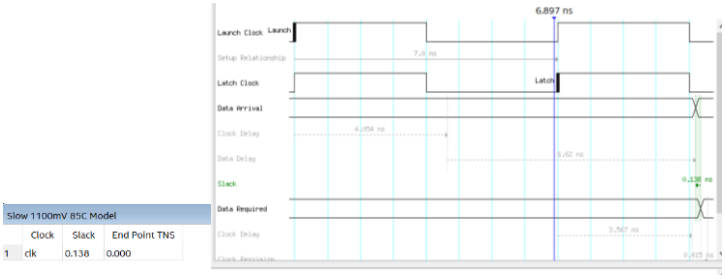
Flow Summary	
Flow Status	Successful - Tue Oct 04 23:39:32 2022
Quartus Prime Version	18.1.0 Build 625 09/12/2018 SJ Lite Edition
Revision Name	cla_clk
Top-level Entity Name	rca_clk
Family	Cyclone V
Device	5CSXFC6D6F31C6
Timing Models	Final
Logic utilization (in ALMs)	31 / 41,910 (< 1 %)
Total registers	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 cla 보다 더 적은 logic utilization을 확인할 수 있다.

C. Timing Analysis

i. cla_clk

1) 조정후



Slow 1100mV 85C Model

	Clock	Slack	End Point TNS
1	clk	0.138	0.000

2) 최대 동작 주파수

Slow 1100mV 85C Model

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

ii. rca_clk

1) 최대 동작 주파수

Slow 1100mV 85C Model

	Fmax	Restricted Fmax	Clock Name	Note
1	131.23 MHz	131.23 MHz	clk	

5. 고찰 및 결론

A. 고찰

이번 실험은 4bit cla와 이를 이용한 32bit cla를 구현하고 기존에 만든 4bit rca를 32bit rca로 구현하여 이 둘을 비교하는 실험이었다. 이전까지의 실험에 비해 사용하는 모듈들도 많고 새로운 Timing analysis를 사용한다는 점이 매우 힘들었다. 개인적인 실수로 w0를 wo로 표기하여 오류를 찾는데 오래 걸린 점도 있었다.

Timing analysis에서 최대 동작 주파수를 구하기 위해 cla32를 clock에 동기화 시켜 cla_clk를 구현하고 이 때 input과 output을 D-flipflop을 이용하여 동기화 하였다. verilog에서는 register을 사용하여 non-blocking으로 코드를 작성했다.

B. 결론

이번 실험을 통해 32bit cla가 32bit rca보다 크기는 더 크지만 더 빠른 속도로 연산함을 볼 수 있다. Cla의 최대 동작 주파수가 rca의 최대 동작 주파수 보다 큰데 이때 주기는 주파수의 역수 이므로 cla가 더 빠른 것을 확인할 수 있다. cla와 rca의 속도 차이는 bit 수가 늘어날수록 더 큰 차이를 보일 것이며 bit수가 적어질수록 미미한 속도차이를 보일 것이다.

6. 참고문헌

1) 공진흥 교수님/컴퓨터공학기초실험2/광운대학교 컴퓨터정보공학부/2022
2) 공영호 교수님/디지털논리회로2/광운대학교 컴퓨터정보공학부/2022