컴퓨터 공학 기초 실험2 보고서

실험제목: Carry Look-ahead Adder (CLA)

실험일자: 2022년 09월 27일 (화)

제출일자: 2022년 09월 28일 (수)

학 과: 컴퓨터정보공학부

담당교수: 공영호 교수님

실습분반: 화요일 0,1,2

학 번: 2019202050

성 명: 이강현

1. 제목 및 목적

A. 제목

Carry Look-ahead Adder (CLA)

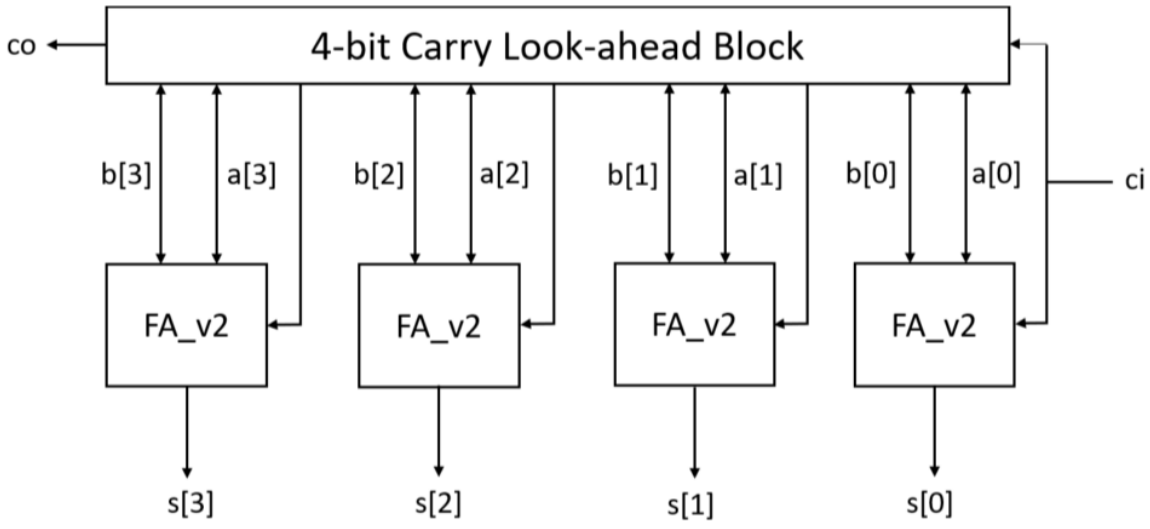
B. 목적

Ripple Carry Adder와 Carry Look-ahead Adder를 비교하여 CLA가 RCA에 비해 어떤 점이 우수한지 알아보고 verilog를 이용하여 설계 및 구현하여 본다. 각 모듈들의 hierarchy가 어떻게 구성되어 있는지 파악하고 Timing Analysis를 통해 RCA와 CLA의 주파수를 비교한다.

2. 원리(배경지식)

<Carry Look-ahead Adder>

CLA은 CLB즉 Carry Look-ahead Block을 사용하여 직렬로 carry를 계산하는 RCA와는 다르게 Block내에서 각자리를 입력받고 carry를 sum과 별도로 예측하여 계산한다.



- 4-bits Carry Look-ahead Adder -

carry-in을 입력으로 가지지 않는 full adder를 사용하며 이 덕분에 full adder가 직렬로 연결되어 뒷자리가 계산되지 않으면 앞자리를 계산할 수 없는 RCA와는 다르게 논리식을 사용하여 처리속도를 줄일 수 있다는 장점을 가진다.

CLB는 다음과 같은 논리식을 가진다.

$$G_i = A_i B_i$$
$$P_i = A_i + B_i$$

Generation(Gi)과 Propagation(Pi)을 full adder의 carry out에 적용하면 아래와 같은 논리식을 도출할 수 있다.

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

따라서 위의 일반식을 4 bit CLA에 적용하면

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

위와 같은 각 자릿수의 carry out을 얻을 수 있다. 위의 식을 사용하여 CLB를 설계할 수 있다.

<32-bit CLA>

32-bit CLA는 4-bit CLA 8개를 직렬로 연결하여 설계한다.

3. 설계 세부사항

<4-bits Carry Look-ahead Adder>

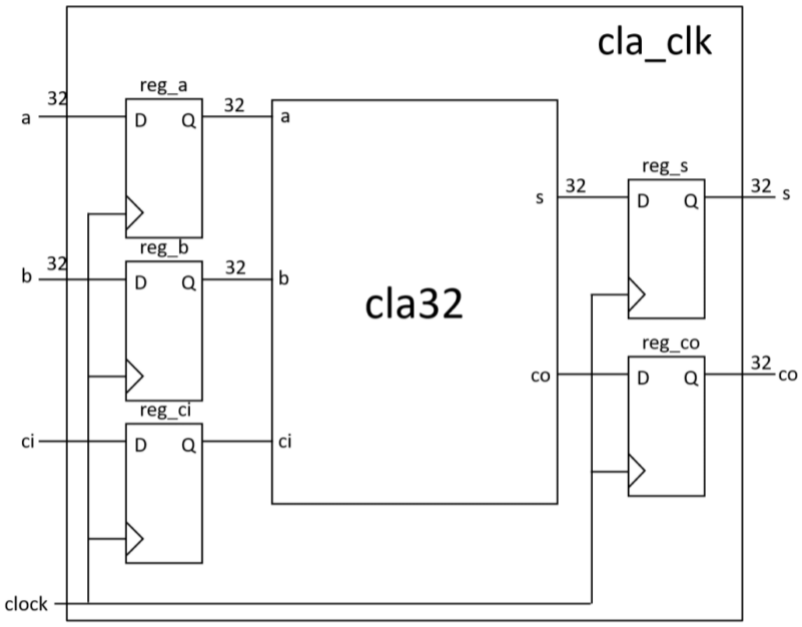
RCA를 구현할 때 사용하였던 gates.v파일내에 CLB를 설계하기 위해 필요한 logic gate인 3,4,5 input and,or gate 모듈을 추가해주었다. 또한 fa.v 기존의 full adder 모듈은 carry in이 필요했지만 이 작업을 CLB에서 해주므로 fa.v를 수정하여 fa_v2로 모듈을 수정하는 과정을 거쳤다.

따라서 CLB는 위에서 설명한 논리식에 따라 carry out을 도출하고 fa_v2는 xor2 모듈을 두개 사용하여 sum값만을 도출한다.

Input			Output
A	B	Cin	S
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

<full adder의 truth table>

<32-bits CLA with Register>



위는 d filpflop을 활용하여 값을 저장하고 클럭을 병렬연결하여 클럭에 따라 값들을 변화할 수 있게끔 설계한 것이다.

cla32는 앞서 구현한 cla4를 8개 직렬 연결하여 구현하였다.

<32-bits RCA with Register>

RCA32는 지난 실습의 rca4모듈들을 cla32와 마찬가지로 직렬 연결하고 레지스터 구현 또한 동일하게 진행하였다.

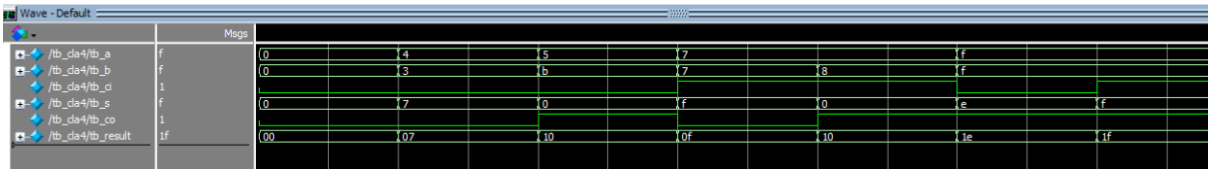
4. 설계 검증 및 실험 결과

A. 시뮬레이션 결과

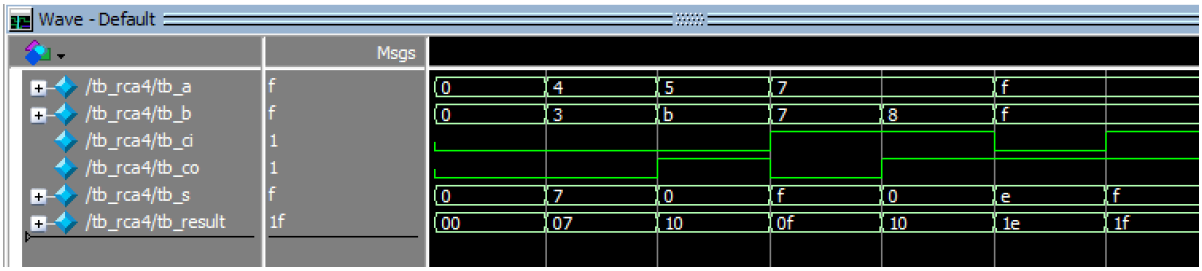
시뮬레이션은 모든 경우를 고려하기엔 너무 많은 입력값을 요구하므로 exhaustive verification이 아닌 directed verification을 사용하였다. 전체적으로 입력값이 모두 0인경우 캐리가 발생하는 경우 캐리가 발생하지 않는 경우, 오버플로우가 발생하는 경우등으로 나누어 입력값을 선정하였다.

<4-bits Carry Look-ahead Adder>

CLA4 시뮬레이션 결과



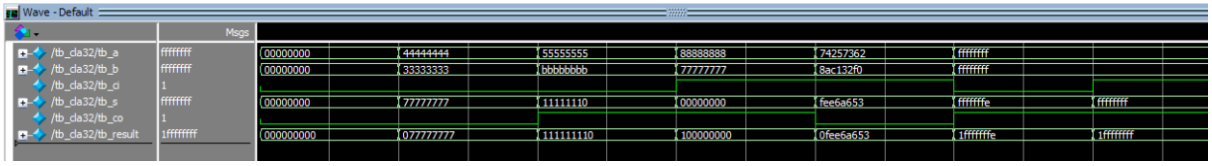
RCA4 시뮬레이션 결과



같은 값을 넣어서 시뮬레이션 돌린 결과 회로구성만 다를뿐 동일한 동작을 하는 것을 확인할 수 있다.

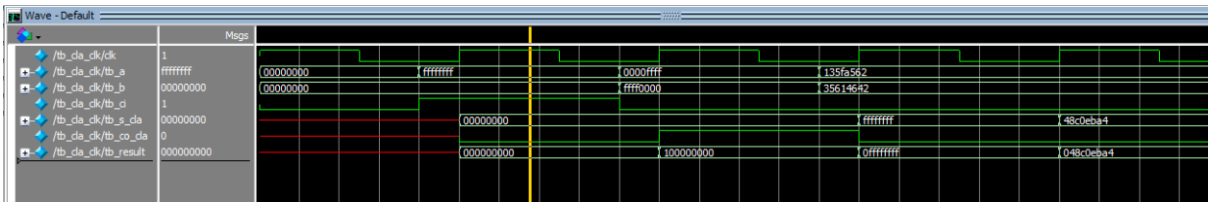
아무 값도 넣지 않은 케이스, 캐리가 발생하지 않는 케이스, 동일한 값을 준 케이스, 캐리가 발생하는 케이스 모두 정상 작동하였다.

<32-bits Carry Look-ahead Adder>



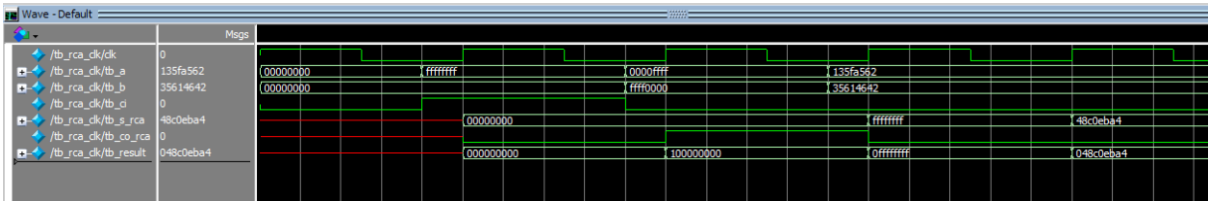
CLA4를 이용하여 만든 CLA32의 시뮬레이션 결과이다. 정확한 값들을 확인할 수 있다.

<32-bits CLA with Register>



Test bench를 통해 16진수로 값들을 넣었고 always 구문을 통해 clock을 구현하였다. 그 결과 clock값에 따라 값이 변하는 것을 확인했다. 하지만 결과값까지 딜레이가 발생하는 것을 확인할 수 있고 이는 클록 주파수 변경을 통해 아래에서 해결할 것이다.

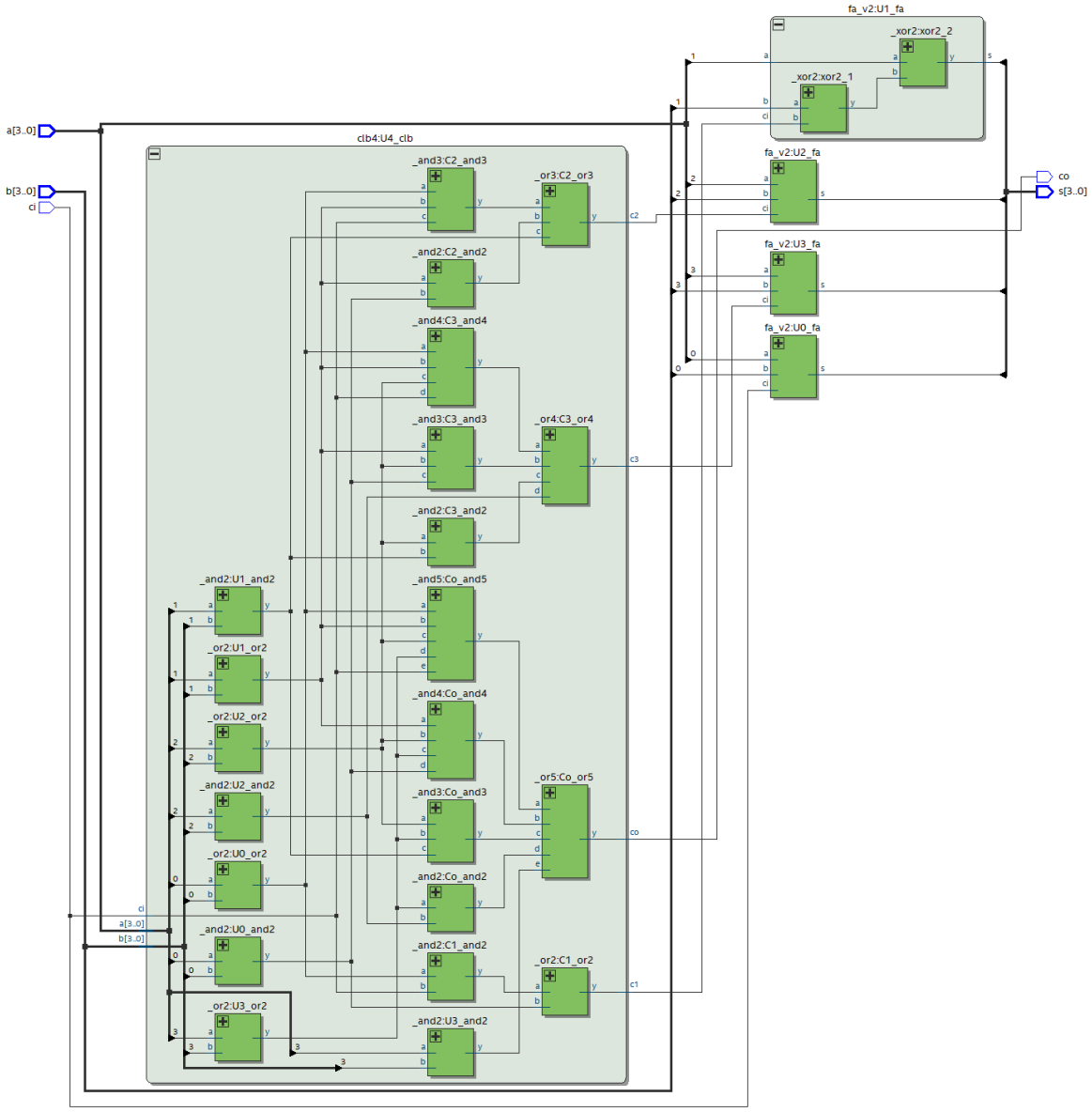
<32-bits RCA with Register>



Rca_clk 또한 같은 값을 테스트벤치를 통해 넣어주었고 그결과 동일한 값을 확인할 수 있다.

B. 합성(synthesis) 결과

<4-bits Carry Look-ahead Adder>



clb블록의 내부 구현도와 전체적인 cla4모듈의 구성을 확인할 수 있다.

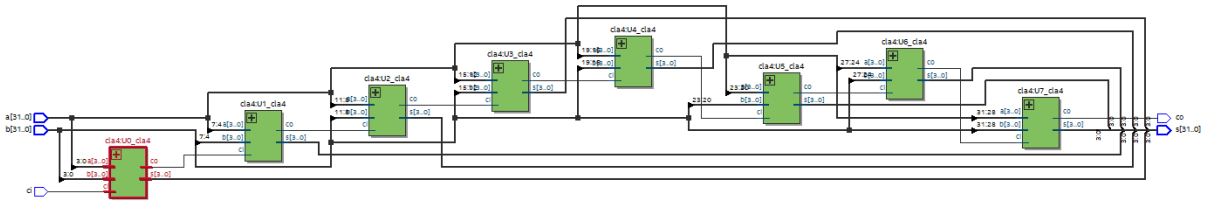
Flow Summary

<<Filter>>

Flow Status	Successful - Wed Sep 28 01:06:53 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)	N/A
Total registers	0
Total pins	14
Total virtual pins	0
Total block memory bits	0
Total DSP Blocks	0
Total HSSI RX PCSs	0
Total HSSI PMA RX Deserializers	0
Total HSSI TX PCSs	0
Total HSSI PMA TX Serializers	0
Total PLLs	0
Total DLLs	0

14개의 핀을 사용하였고 정상적인 결과를 확인할 수 있다.

<32-bits Carry Look-ahead Adder>



Cla4 모듈들이 직렬로 연결되어 있음을 확인한다.

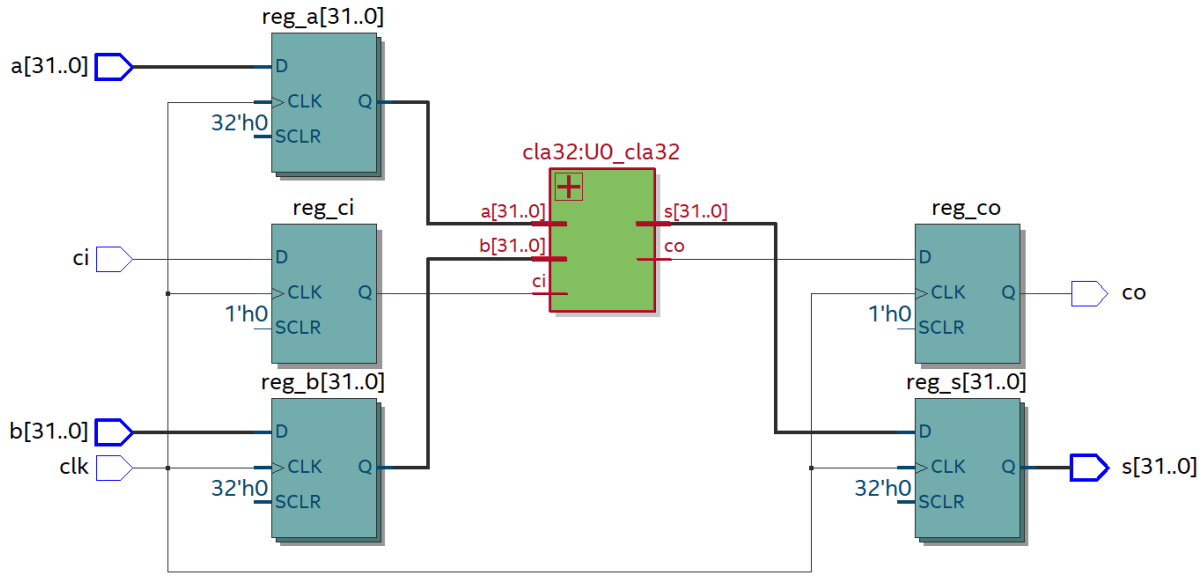
Flow Summary

<<Filter>>

Flow Status	Successful - Wed Sep 28 01:14:03 2022
Quartus Prime Version	18.1.0 Build 625 09/12/2018 SJ Lite Edition
Revision Name	cla_clk
Top-level Entity Name	cla32
Family	Cyclone V
Device	5CSXFC6D6F31C6
Timing Models	Final
Logic utilization (in ALMs)	N/A
Total registers	0
Total pins	98
Total virtual pins	0
Total block memory bits	0
Total DSP Blocks	0
Total HSSI RX PCSs	0
Total HSSI PMA RX Deserializers	0
Total HSSI TX PCSs	0
Total HSSI PMA TX Serializers	0
Total PLLs	0
Total DLLs	0

Cla32를 구현하는데 사용된 핀개수는 98개이며 정상적인 결과이다.

<32-bits CLA with Register>



cla_clk의 구성도이다. 입력과 출력에 레지스터들이 배치되어 있고 클럭이 병렬로 잘 연결되어 있는 모습을 확인할 수 있다.

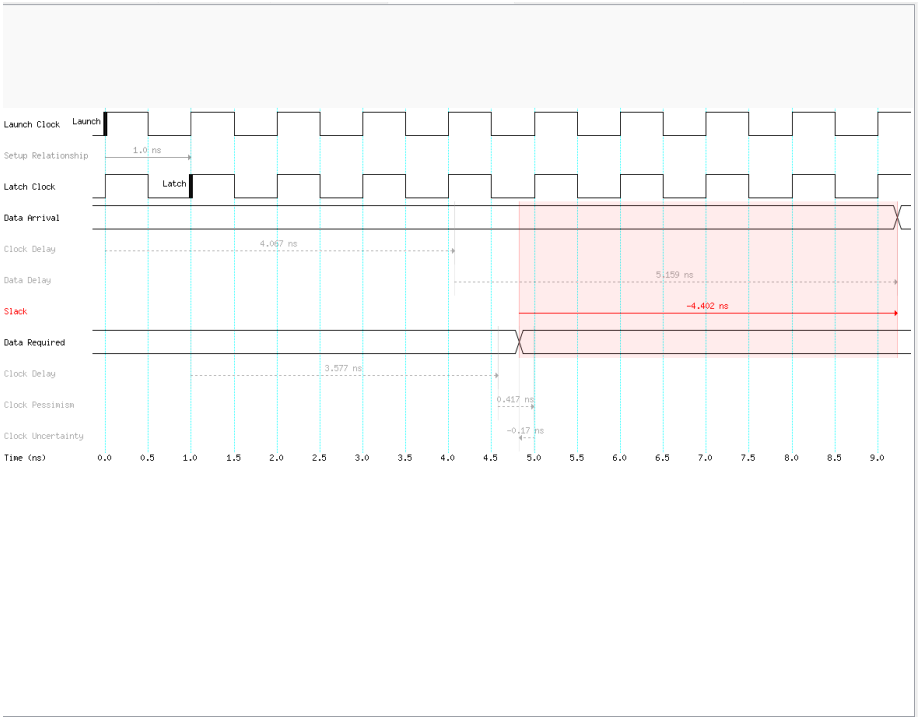
Flow Summary

<<Filter>>

Flow Status	Successful - Wed Sep 28 01:24:36 2022
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)	43 / 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 %)

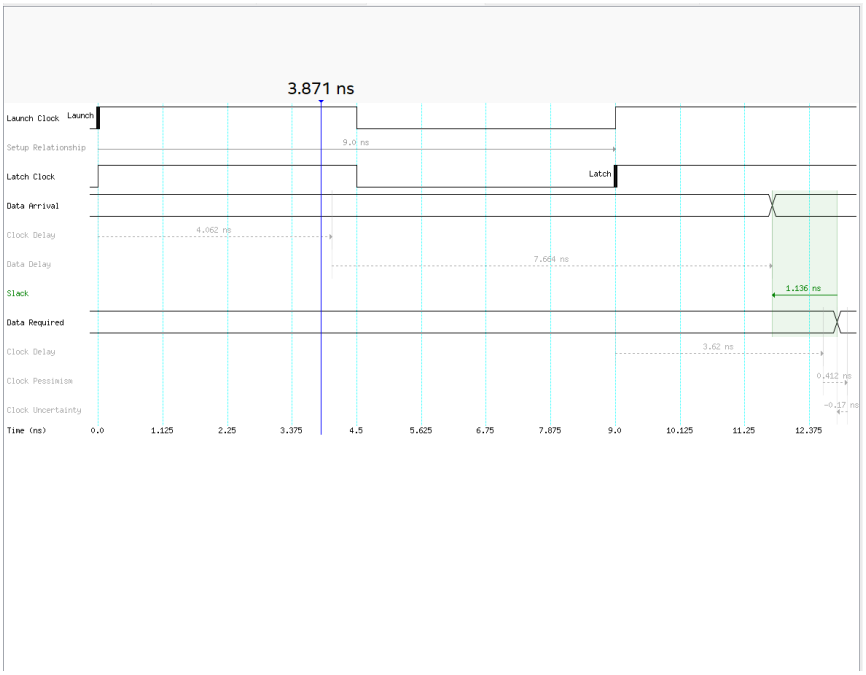
사용된 logic elements들의 개수와 pin개수를 확인할 수 있다.

<Timing Analysis>



위의 시뮬레이션 결과에서 설명했듯이 결과값에 딜레이가 발생하였고 slack이 음수임은 violation을 야기하므로 clock값을 조절해야함을 알 수

있다.



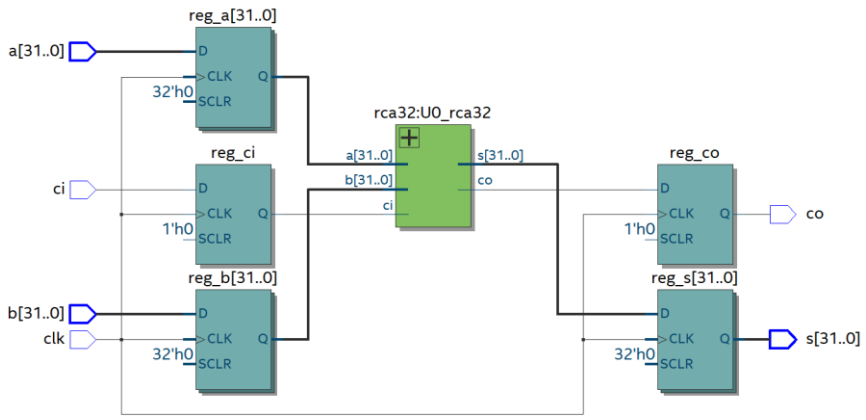
따라서 clock을 9ns로 조절하여 slack값을 양수로 바꿔주었다. 따라서 violation은 발생하지 않았다.

Slow 1100mV 85C Model

	Fmax	Restricted Fmax	Clock Name	Note
1	127.16 MHz	127.16 MHz	clk	

최대 동작 주파수가 127.18MHz임을 확인할 수 있다.

<32-bits RCA with Register>



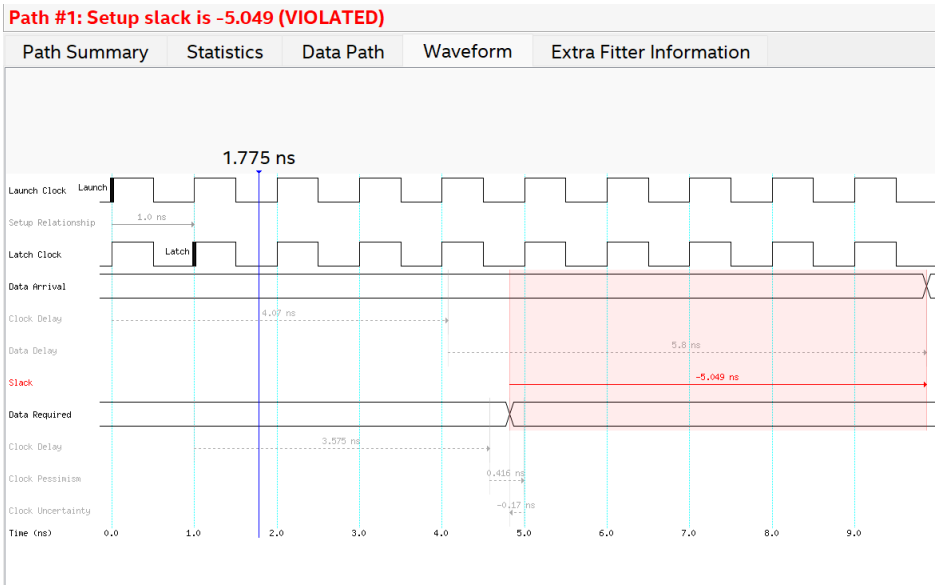
rca_clk또한 입력과 출력에 각각 레지스터를 배치하였고 클럭에 의해 값이 전달되게끔 동작됨을 확인할 수 있다.

Flow Summary

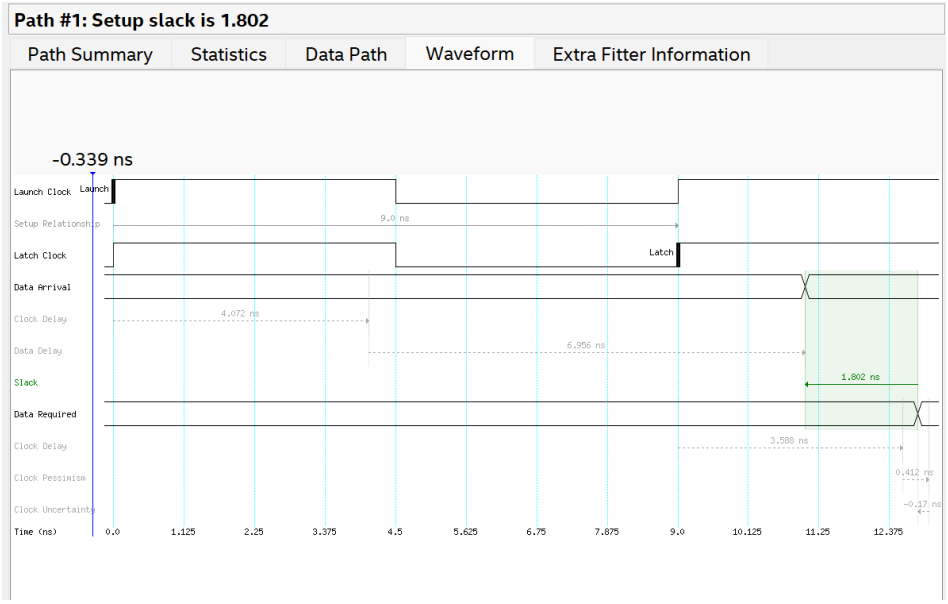
<<Filter>>

Flow Status	Successful - Wed Sep 28 02:39:57 2022
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)	32 / 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 %)

사용된 Logic element의 수와 pin개수를 알 수 있으며 정상적인 결과를 보인다.



rca_clk또한 slack값이 음수라 violation이 발생함을 확인할 수 있다. 따라서 clock값을 조절하여야 한다.



Clock값을 9ns로 수정한 결과 slack값이 양수가 되면서 더 이상 violation은 발생하지 않았다.

Slow 1100mV 85C Model

	Fmax	Restricted Fmax	Clock Name	Note
1	138.93 MHz	138.93 MHz	clk	

최대 동작 주파수가 138.93 MHz임을 확인했다.

5. 고찰 및 결론

A. 고찰

지난 시간에 배운 half adder와 full adder로 구현한 rca에서 수정하여 cla를 구현해보았다. 기존에 배웠던 내용들의 응용이라 이해하는데 어려움은 없었지만 always 구문을 통해 clock을 구현하는 방법과 그를 통해 timing analysis 기법을 활용하여 회로를 분석하는 방법과 같이 새로운 내용들이 많이 추가가 되어 헷갈리는 부분과 낯선 부분들이 많이 있었다. 또한 =과 <=의 차이와 실제 하드웨어적으로 구현하기 어려운 것들이 쿼터스에서 쉽게 구현가능하다는 점은 유용했다.

B. 결론

이번 실험에서는 결론적으로 RCA보다 carry out의 논리식을 활용하여 처리속도를 높여 계산할 수 있는 CLA를 만들어보면서 이것이 얼마나 효율적으로 동작하는지에 대해 확인해보고 slack과 같은 문제가 발생하였을 때 어떻게 해결할 것인지를 배우고 같은 동작을 하는 회로지만 효율성을 더욱 높이는 것이 중요하다는 것을 인지할 수 있었다. CLA와 RCA에 대한 결과를 FMAX 즉, 최대 주파수가 얼마나 나오는지에 대해 효율성을 검사할 수 있는데 여기서 한가지 궁금한 점이 발생했다. 클럭주기를 9ns로 동일하게 수정하였을 때는 위와 같이 RCA의 주파수가 CLA의 주파수보다 높게 측정이 되면서 RCA가 더 빠르게 작동하였다. 하지만 7ns로 수정하였을 때는 아래와 같이

Slow 1100mV 85C Model				
	Fmax	Restricted Fmax	Clock Name	Note
1	155.33 MHz	155.33 MHz	clk	

<RCA>

Slow 1100mV 85C Model				
	Fmax	Restricted Fmax	Clock Name	Note
1	171.38 MHz	171.38 MHz	clk	

<CLA>

CLA가 더 높은 성능을 보여주었다.

이는 slack이 발생하지 않을 정도의 클럭주기만 주어진다면 CLA가 더욱 높은 성능을 내는 것으로 확인되었다.

클럭주기를 최대한 줄이는 것이 하드웨어적으로 전체 속도를 상승시키는 것이므로 CLA가 더 성능이 높은 것으로 결론지을 수 있겠으나 그거와 별개로 항상 CLA가 높은 성능을 내지 않는다는 것은 의문점이었다.

<modified CLA 구현>

Clb 부분 코드를 ppt내의 구조로 표현하여 carryout을 계산을 먼저 계산하면서 and or and or 이 반복되는 각 게이트에서 c1,c2,c3를 뽑아내는 방법을 통해 구현하였다. Violation이 발생했기에 똑같이 클럭을 7ns로 수정하였다.

```
_and2 C1_and2(p[0],ci,w0_c1);
_or2 C1_or2(w0_c1,g[0],c1);
//c1 instance

_and3 C2_and3(p[0],p[1],ci,w0_c2);
_or2 C2_or2(w3,w0_c2,c2);
//c2 instance

_and4 C3_and4(p[0],p[1],p[2],ci,w0_c3);
_or2 C3_or2(w5,w0_c3,c3);
//c3 instance

_and4 U0(p[0],p[1],p[2],p[3],w0);
_and2 U1(w0,ci,w1);
_and2 U2(g[0],p[1],w2);
_or2 U3(w2,g[1],w3);
_and2 U4(w3,p[2],w4);
_or2 U5(w4,g[2],w5);
_and2 U6(w5,p[3],w6);
_or2 U7(w6,g[3],w7);
_or2 U8(w7,w1,co);
//co instance
```

modified CLA의 CLB부분

기존에 논리식을 그대로 사용하여 하나하나 계산했을때보다 게이트 통과 수가 줄어서 아래와 같이 동작주파수가 개선된 모습을 확인했다.

Slow 1100mV 85C Model

	Fmax	Restricted Fmax	Clock Name	Note
1	189.0 MHz	189.0 MHz	clk	

6. 참고문헌

공영호 교수님/컴퓨터공학기초실험2/2022