컴퓨터 공학 기초 실험 2

Assignment 3. Carry Look-ahead Adder (CLA)

1. Carry Look-ahead Adder

- Functional Description

✓ Ripple carry adder (RCA)가 계산이 완료될 때까지 시간이 많이 걸리는 단점을 보완하기 위해 입력 a, b 그리고 carry in 이 주어질 때, 모든 올림수가 동시에 구해져 계산시간을 단축시키는 가산기이다.

✓ Carry (올림 수)를 계산하기 위해 carry 만을 계산해주는 별도의 carry look-ahead block (CLB)이 존재한다.

✓ 본 실습에서는 full adder 와 4-bits carry look ahead block 을 사용하여 4-bit carry look-ahead adder 를 구현하여 본다.

- Carry Look-ahead Block Boolean Equation

✓ Carry look-ahead block 으로 carry out 값을 미리 계산해주기 위해서는 generation signal(G_i), propagation signal(P_i)을 아래와 같이 정의한다.

$$G_i = A_i B_i$$

$$P_i = A_i + B_i$$

✓ 위 식을 full adder 의 carry out 에 적용하면 다음과 같다.

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

✓ 이를 적용하여 4-bits CLA 를 위한 carry 를 미리 계산하면 다음과 같다.

$C_1 \quad =$

$C_2 \quad =$

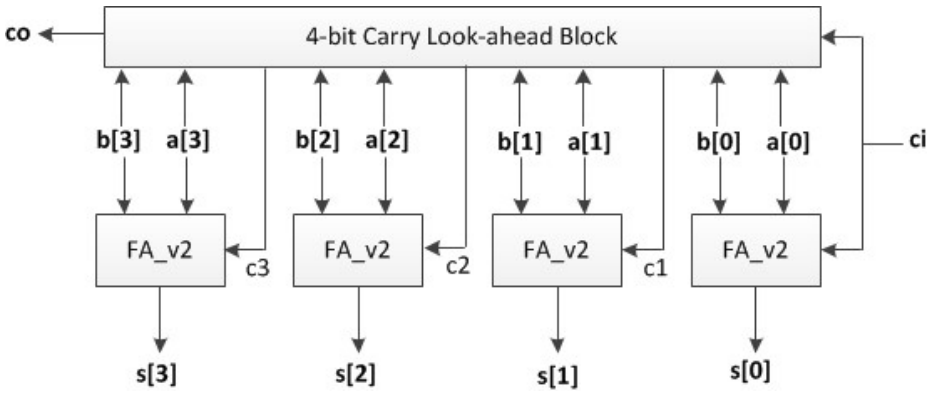
$C_3 \quad =$

$Cout =$

(직접 boolean equation 을 작성하여 레포트에 정리한다.)

- 4-bits Carry Look-ahead Adder Structural Description

✓ 다음은 4-bits CLA 의 structural description 을 나타낸다.



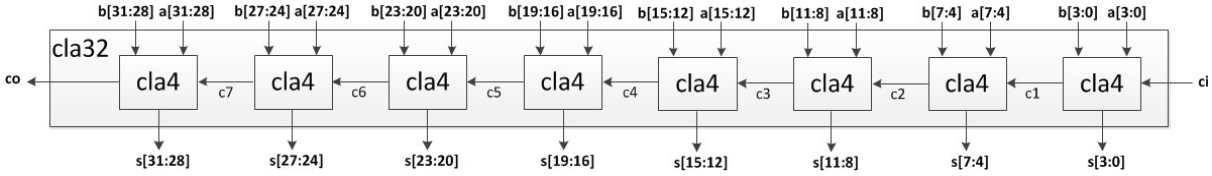
.

- 32-bits Carry Look-ahead Adder

✓ 32-bits CLA 는 1-bit carry in 와 4-bit 입력 a, b 를 받아 4-bit 출력 sum 와 1-bit carry out 을 발생시키는 4-bits CLA 8 개를 직렬로 연결하여 32-bits CLA 를 구현한다.

- 32-bits Carry Look-ahead Adder Structural Description

✓ 다음은 32-bits CLA 의 structural description 을 나타낸다.



2. 32-bits CLA with clock

➢ Functional Description

✓ 앞선 32-bits CLA 에 앞, 뒤로 flip-flop 을 추가하여 구현하며, 해당 flip-flop 에 clock 을 연결하여, 32-bits CLA 의 valid 한 결과가 나오는 데 필요한 최대 동작 주파수를 찾는 것을 목적으로 한다.

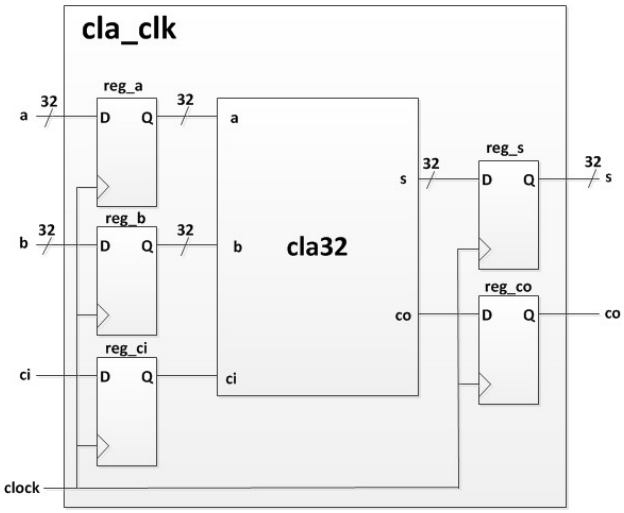
- Flip-flop 에 대해서는 다음 이론 및 실습을 통해 배우기로 한다.

✓ CLA 의 delay 를 확인하기 위하여 CLA 에 앞, 뒤로 flip-flop 을 삽입하여 circuit 의 clock pried 를 확인한다.

- Register transfer level(RTL)의 combination circuit elements 는 delay 를 포함하고 있지 않아 CLA 단독의 delay 를 확인할 수 없다.

➢ Structural Description

✓ cla_clk module 의 입력과 출력은 D-flipflop 과 연결되어 있으며 모든 D-flipflop 은 입력 clk 에 연결 되어있다.



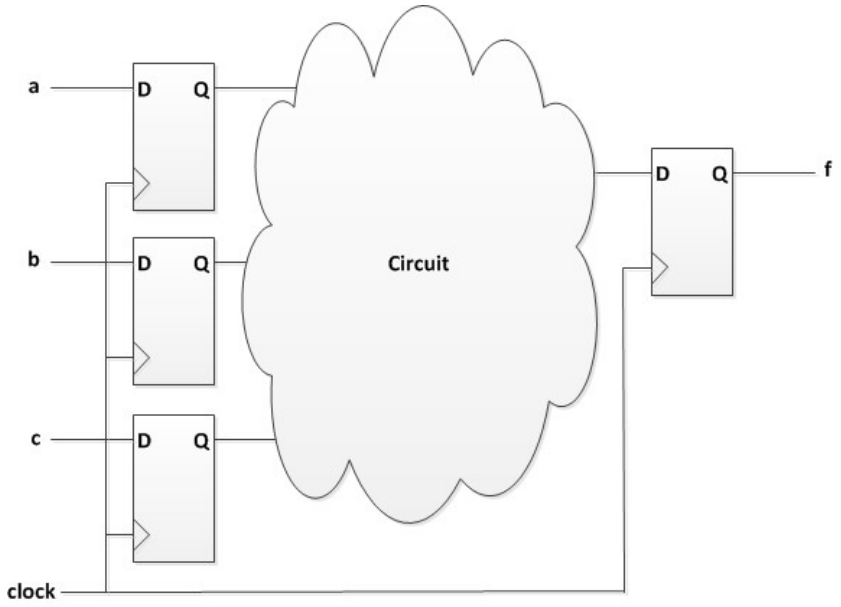
3. Timing Analysis

➢ Timing Analysis

✓ A process of analyzing delays in a logic circuit to determine the conditions under which the circuit operates reliably.

✓ These conditions include, but are not limited to, the maximum clock frequency (f_{max}) for which the circuit will produce a correct output.

✓ 즉, timing analysis 는 구현한 logic 이 제대로 동작할 수 있는 조건을 찾기 위하여 해당 circuit 의 delay 를 분석하는 과정이다. 일반적으로 여기서 찾는 조건은 최대 동작 주파수(maximum clock frequency)이다.



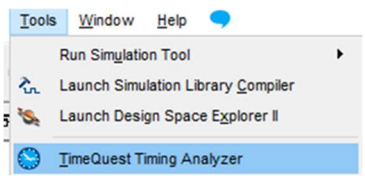
➢ Timing Analysis lab

✓ 해당 예시는 `cla_clk` 모듈을 이용한다.

✓ 주의사항

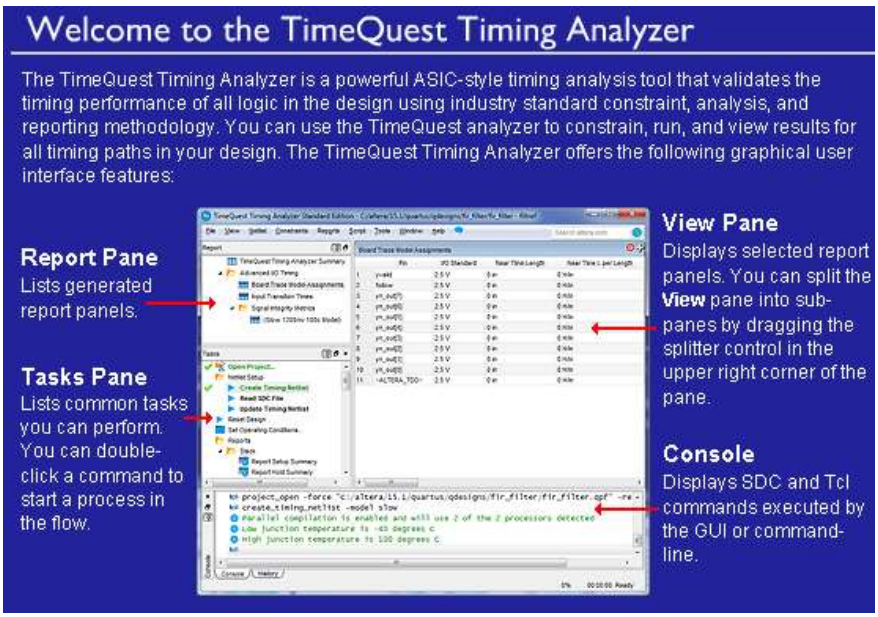
- `cla` 모듈을 이용하면 해당 과정이 정상적으로 수행할 수 없음
- Quartus II 의 "Compile design" 중 "Timing analysis" 과정을 수행해야함

✓ Tools ➔ TimeQuest Timing Analyzer 를 선택



✓ 아래의 사진은 TimeQuest 의 pane 에 대해 기술되어 있다. TimeQuest 의 여러 pane 중에서 주로 우측에 있는 report pane 과 tasks pane 을 사용하는 데, report pane 은 tool 에서 생성된 report 의 list 를 보여주며, tasks pane 은 timing reports 를 얻기 위하여

수행하여야 하는 action 들의 순서를 보여주고 있다. 이중, tasks pane 의 'Create Timing Netlist', 'Read SDC File', 'Update Timing Netlist'를 순서대로 double click 한다.



- 이때 기본으로 설정되는 clock 은 1ns 이다.

➢ Task pane 의 'Report Setup Summary'를 double click 한다. 그러면 하나의 창을 통해 'slack'과 'End Point TNS(Total Negative Slack)'를 확인할 수 있다.

- 여기서 slack 은 destination flip-flop 에 도착해야 되는 시간과 실제 도착 시간의 시간 차로써, 시간 차가 음수로 나오면 violation 이 발생할 수 있다.
- 아래의 그림을 확인해 보면 구현한 32-bit CLA 의 경우, -3.496 의 slack 을 가지고 있으므로 이에 대한 clock 조정이 필요하다.
  - slack, TNS 등의 timing 은 상이할 수 있다.

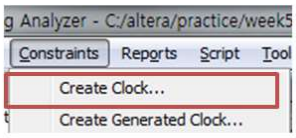
✓ 이를 더 자세히 보기위해 마우스 우클릭 후 'Report Timing'을 선택하며, 'Setup: clock'이라는 Window 가 뜬다.

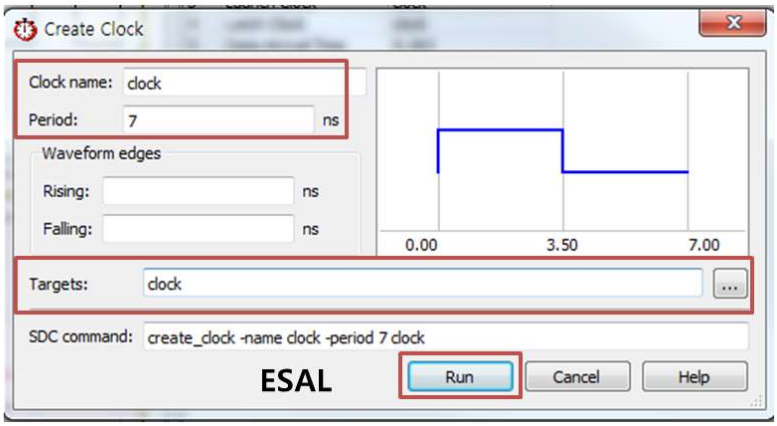
- Window 는 3 가지 부분으로 구현되어 있으며, 제일 상단은 slack 시간이긴 상위 10 개의 path 를 나타내 주며, 아래는 그에 대한 세부정보를 보여준다. 실습에서 waveform 을 선택하도록 한다.

- Waveform 을 보면 빨간 색으로 slack 이라고 뜬 부분을 통해 data 가 필요한 시점보다 -3.496ns 늦게 도착한다는 것을 확인할 수 있다.
- Waveform 에서 data arrival 은 실제 데이터가 도착하는 시간이고, data required 는 data 가 필요한 시점이다.

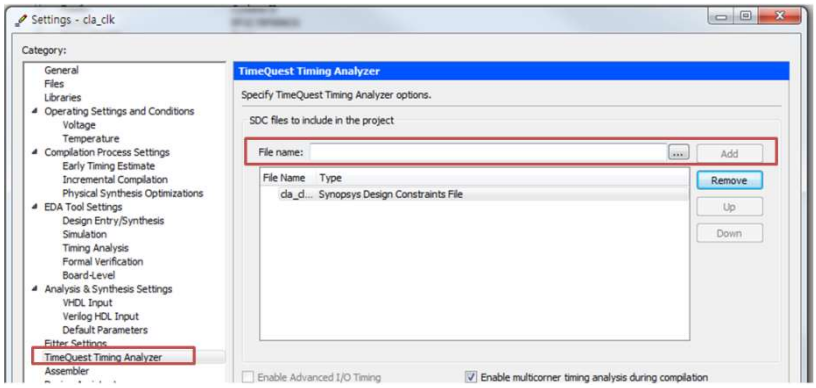
✓ Clock 을 조절해주기 위하여 'Constraints ➔ Create Clock'을 선택

- 현재 주기가 1ns 일 때, -3.496ns 의 slack 발생하였으므로 clock 주기를 7 로 설정한 후 Run 을 누른다.
- Clock name 은 cla_clk 의 clock port name 을 기술한다.





- 다음으로 이를 SDC fie 에 저장하기 위하여 task pane 의 'Write SDC File'을 선택한다. 이후 TimeQuest 를 끄고, 방금 생성한 SDC file 을 적용하기 위하여 Quartus 에서 'Assignments ➔ Setting'을 들어가 TimeQuest Timing Analyzer 를 선택하여 file 을 추가한다.



- 다시 컴파일한 후에 TimeQuest 를 들어가서 앞서 한 것처럼 slack 을 확인하여 본다.

✓ Tasks pane 의 'Report Fmax Summary'를 double click 하여 해당 circuit 의 최대 동작 주파수를 확인하여 본다.

➢ 32-bits CLA with Register and 32-bits RCA with Register

✓ 앞선 32-bits CLA with Register 와 유사한 방법으로 수행한다.

4. Verilog 구현

- ➢ 해당 챕터에 기술되지 않은 모듈은 이전 과제와 및 강의자료를 참고하여 구현
- ➢ Design specification
- ✓ Module hierarchy description
  - 기술된 top module 과 sub module 의 이름은 표와 **반드시 동일**해야 한다.
  - 표에 나와 있지 않은 sub module 은 이전 과제를 참고한다.
  - 아래의 표는 `cla_clk` 의 hierarchy 를 나타낸다.

구분	이름	설명
Top module	cla_clk	32 bits CLA with register
Sub module	cla32	32 bits CLA
Sub module	cla4	4 bits CLA with clb module
Sub module	clb4	4 bits carry look-ahead block
Sub module	fa_v2	Full adder without carry out port

  - 아래의 표는 `rca_clk` 의 hierarchy 를 나타낸다.
  - 이전 과제에 사용한 rca 모듈을 사용하여 `rca_clk` 구현

구분	이름	설명
Top module	rca_clk	32 bits RCA with register
Sub module	rca32	32 bits RCA
Sub module	rca4	4 bits RCA
Sub module	fa	Full adder

- ➢ File configuration
- ✓ cla_clk.v – Carry lock-ahead adder with register 구현
- ✓ cla32.v – module `cla4` 를 instance 하여 구현
- ✓ cla4.v – full adder 와 clb 를 instance 하여 4 bits CLA 구현
- ✓ clb4.v – carry look-ahead block 구현
- ✓ rca_clk.v –Ripple carry adder with register 구현
- ✓ rca32.v – 이전 과제 module `rca4` 를 instance 하여 구현
- ✓ fa_v2.v – fa_v2 module 구현

4.1. Full adder without carry out port

- 기존의 full adder 의 carry out 이 필요가 없기 때문에 sum 만 출력하도록 이전의 full adder module 을 수정
- Module specification
- ✓ Module name: fa_v2
- ✓ I/O configuration
  - I/O 는 표와 **반드시 동일**해야 한다.
  - wire/reg 의 경우 자유롭게 추가, 삭제가 가능하다.

구분	이름	비트 수	설명
Input	a	1 bit	Input data a
	b	1 bit	Input data b
	ci	1 bit	Carry in
Output	s	1 bit	Sum

4.2. Carry look-ahead block

- Module specification
- ✓ Module name: clb4
- ✓ File name: clb4.v
- ✓ I/O configuration
  - I/O 는 표와 **반드시 동일**해야 한다.
  - wire/reg 의 경우 자유롭게 추가, 삭제가 가능하다.
  - 다음 표에 나온 wire/reg 는 예시이다.

구분	이름	비트 수	설명
Input	a	4 bits	Input data A
	b	4 bits	Input data B
	ci	1 bit	Carry in
Output	c1	1 bit	Carry of bit position [0]
	c2	1 bit	Carry of bit position [1]
	c3	1 bit	Carry of bit position [2]
	co	1 bit	Carry out

4.3. 4 bits carry look ahead adder

- ➢ Module specification
- ✓ Module name: cla4
- ✓ File name: cla4.v
- ✓ I/O configuration
  - I/O 는 표와 **반드시 동일**해야 한다.
  - wire/reg 의 경우 자유롭게 추가, 삭제가 가능하다.
  - 다음 표에 나온 wire/reg 는 예시이다.

구분	이름	비트 수	설명
Input	a	4 bits	Input data a
	b	4 bits	Input data b
	ci	1 bit	Carry in
Output	co	1 bit	Carry out
	s	4 bits	Sum
Wire	c	3 bits	Carry wire

- ✓ Instance module description
  - 다음 표에 나온 모듈의 instance name 은 예시이다.

Classification	Name	Description
Module	cla4	4-bit carry look-ahead adder
Sub module	U0_fa_v2	Full adder
	U1_fa_v2	
	U2_fa_v2	
	U3_fa_v2	
	U4_clb	Carry generation

4.4. 32 bits carry look ahead adder

- ➢ Module specification
- ✓ Module name: cla32
- ✓ File name: cla32.v
- ✓ I/O configuration
  - I/O 는 표와 **반드시 동일**해야 한다.
  - wire/reg 의 경우 자유롭게 추가, 삭제가 가능하다.
  - 다음 표에 나온 wire/reg 는 예시이다.
  - 내부 carry 의 경우, 8-bit wire 를 사용하여도 되고, 1-bit wire 를 8 개 사용하여도 무관하다.

Port	Name	Bandwidth	Description
Input	a	32 bits	Input data A
	b	32 bits	Input data B
	ci	1 bit	Carry in
Output	co	1 bit	Carry out
	s	32 bits	Sum
Wire	c	8 bits	Internal carry

- ✓ Instance module description
  - cla4 module 을 8 개 instance 한다.
  - 다음표에 나온 모듈의 instance name 은 예시이다.

Classification	Name	Description
Module	cla32	32-bit carry look-ahead adder
Instance	U0_cla4	4-bit CLA
	U1_cla4	
	U2_cla4	
	U3_cla4	
	U4_cla4	
	U5_cla4	
	U6_cla4	
	U7_cla4	

4.5. Carry lock-ahead adder with register

- Module name: cla_clk
- File name: cla_clk.v
- ✓ I/O configuration
  - I/O 는 표와 **반드시 동일**해야 한다.
  - wire/reg 의 경우 자유롭게 추가, 삭제가 가능하다.
  - 다음 표에 나온 wire/reg 는 예시이다.
  - wire/reg 의 경우 자유롭게 추가, 삭제가 가능하다.

구분	이름	비트 수	설명
Input	clk	1 bit	Clock
	a	32 bits	Input data A
	b	32 bits	Input data B
	ci	1 bit	Carry in
Output	co	1 bit	Carry out
	s	32 bits	Sum
reg	reg_a	32 bits	Registers of input a
	reg_b	32 bits	Registers of input b
	reg_ci	1 bit	Registers of carry in
	reg_s	32 bits	Registers of sum
	reg_co	1 bit	Registers of carry out

4.6. 32 bits ripple carry adder

- Module name: rca32
- File name: rca32.v

✓ I/O configuration

- I/O 는 표와 **반드시 동일**해야 한다.
- wire/reg 의 경우 자유롭게 추가, 삭제가 가능하다.
- 다음 표에 나온 wire/reg 는 예시이다.
- wire/reg 의 경우 자유롭게 추가, 삭제가 가능하다.

Port	Name	Bandwidth	Description
Input	a	32 bits	Input data A
	b	32 bits	Input data B
	ci	1 bit	Carry in
Output	co	1 bit	Carry out
	s	32 bits	Sum
Wire	c	8 bits	Internal carry

✓ Instance module description

- rca4 module 을 8 개 instance 하거나 full adder 32 개를 instance 한다.

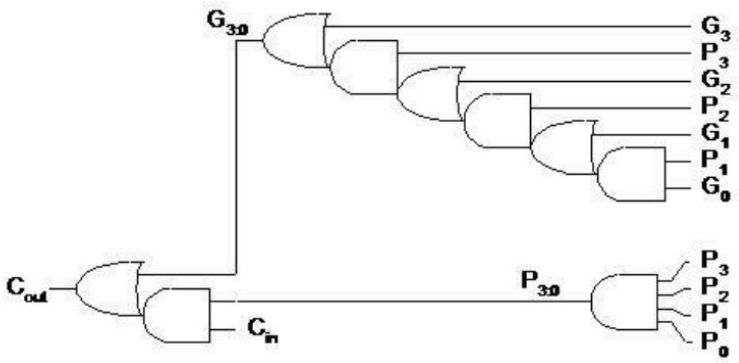
4.7. Ripple carry adder with register

- Module name: rca_clk
- File name: rca_clk.v
- ✓ I/O configuration
  - I/O 는 표와 **반드시 동일**해야 한다.
  - wire/reg 의 경우 자유롭게 추가, 삭제가 가능하다.
  - 다음 표에 나온 wire/reg 는 예시이다.
  - wire/reg 의 경우 자유롭게 추가, 삭제가 가능하다.

구분	이름	비트 수	설명
Input	clk	1 bit	Clock
	a	32 bits	Input data A
	b	32 bits	Input data B
	ci	1 bit	Carry in
Output	co	1 bit	Carry out
	s	32 bits	Sum
reg	reg_a	32 bits	Registers of input a
	reg_b	32 bits	Registers of input b
	reg_ci	1 bit	Registers of carry in
	reg_s	32 bits	Registers of sum
	reg_co	1 bit	Registers of carry out

5. Report

- ➢ 레포트는 공지사항에 올린 보고서 양식에 맞추어 작성하고, 다음의 사항에 대하여서도 추가적으로 작성한다.
- ✓ RTL viewer, flow summary 를 포함할 것
- ✓ Waveform 을 통한 검증의 경우, RCA 에서 검증했던 내용을 참고하여 진행
  - Functional simulation 의 결과는 같기 때문에 RCA 와 CLA 를 같은 testbench 를 사용하여 결과가 같다는 것을 보여줘도 된다.
- ➢ Timing analysis 결과
- ✓ (Setup: clock 에서 waveform 및 Fmax 삽입)
- ✓ 결론 및 고찰에 32-bits CLA 와 32-bits RCA 의 크기 및 속도를 비교할 것
- ✓ 아래 구조와 같은 modified 32-bits CLA 를 구현하고, 앞서 구현한 32-bits CLA 와 size 및 operation frequency 를 비교하여 본다. 자세한 내용은 디지털논리회로 2 강의 자료를 참고



- ➢ 채점기준

세부사항		점수	최상	상	중	하	최하
소스코드	Source code 가 잘 작성되었는가? (Structural design 으로 작성되었는가?)	10	10	8	5	3	0
	주석을 적절히 달았는가? (반드시 영어로 주석 작성)	20	20	15	10	5	0
설계검증 (보고서)	보고서를 성실히 작성하였는가? (보고서 형식에 맞추어 작성)	30	30	20	10	5	0
	합성결과를 설명하였는가?	10	10	8	5	3	0
	검증을 제대로 수행하였는가? (모든 입력 조합, waveform 설명)	30	30	20	10	5	0
총점		100					

6. Submission

➢ 제출기한

- 자세한 제출기한은 KLAS 와 일정을 참고

➢ 과제 업로드

✓ Source code 와 report 를 같이 ZIP 파일로 압축하여 KLAS(종합정보서비스) 과제 제출에 해당 과제 upload

✓ 업로드 파일명은 (요일#)_(학번)_Assignment_#.zip

- 요일번호
  - 실습 미수강은 0
  - 월요일 0, 1, 2 교시 1
  - 화요일 0, 1, 2 교시 2
  - 수요일 0, 1, 2 교시 3
- Ex) 월요일 반 수강, 2019110609, Assignment 1 제출 시 1_2019110609_Assignment_01.zip 으로 제출

✓ Report 명은 (요일#) _(학번) _Assignment_#.pdf

- 요일 번호는 위의 업로드 파일명과 동일하게 진행

✓ Ex) 수요일 반 수강, 2019110609, Assignment 1 제출 시 3_2019110609_Assignment_01.pdf 으로 제출

✓ Report 는 PDF 로 변환해 제출 (미수행시 감점)

➢ 생성한 프로젝트 폴더에 v 파일과 보고서 파일 포함하여 압축

✓ 자동 생성된 output_files 폴더, qpf, qsf 파일 반드시 포함

✓ db, incremental_db, simulation 폴더, ~.bak 파일 삭제 (미수행시 감점)

✓ cla_clk 프로젝트에 cla_clk.v, cla4.v, cla32.v, clb4.v, fa_v2.v, fa.v, gates.v, ha.v, rca_clk.v, rca4.v, rca32.v, tb_cla_clk.v, tb_rca_clk.v, tb_cla32.v, tb_cla4.v 를 함께 프로젝트 파일에 추가하여 한 폴더에 제출한다.