컴퓨터 공학 기초 실험2 보고서

실험제목: Ripple Carry Adder

실험일자: 2022년 09월 21일 (수)

제출일자: 2022년 09월 28일 (수)

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

- Ripple-Carry Adder (RCA)

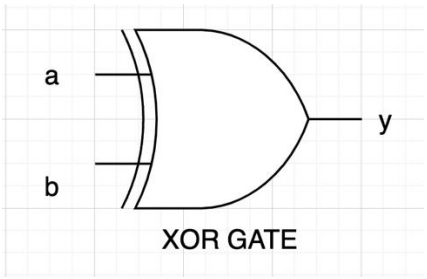
B. 목적

- 각 게이트 모듈을 Verilog로 구현해 본다. 특히 XOR gate를 Karnaugh map을 통해 분석하여 구현한다. Half_Adder, Full_Adder 모듈을 위와 동일한 방법으로 구현하고, 구현한 Adder들을 이용하여 4bit 덧셈 연산을 수행하는 Ripple-Carry Adder를 직접 gate부터 module 단위로 설계한다.

2. 원리(배경지식)

A. XOR GATE

■ XOR 게이트의 기호와 진리표

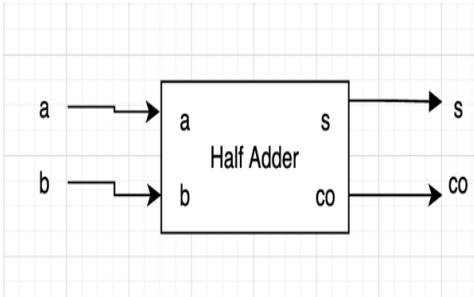


INPUT		OUTPUT
a	b	y
0	0	0
0	1	1
1	0	1
1	1	0

배타적 논리함이라고도 불리는 XOR GATE는 2개의 입력인 경우 서로 다른 2개의 입력이 있을 시 1을 출력하는 gate이다. 기호로는 $y = a \oplus b$라고 표현한다.

B. Half Adder

■ Half Adder의 기호와 진리표

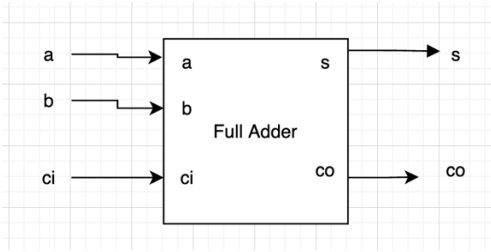


INPUT		OUTPUT	
a	b	s	co
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Half Adder 즉 반가산기는 1bit의 두개의 값 a,b를 입력 받아 가산하여, 합(sum)을 뜻하는 s와 carry out을 뜻하는 co를 출력한다.

C. Full Adder

- Full Adder의 기호와 진리표

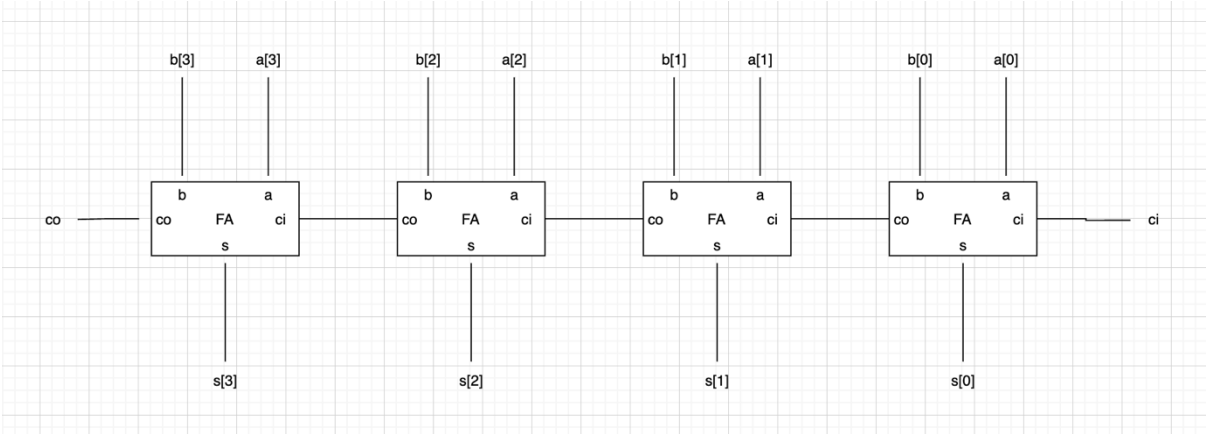


INPUT			OUTPUT	
a	b	ci	s	co
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Full Adder는 가산기라고도 하며, Half Adder에서 ci라는 입력이 추가된 것으로 1bit의 2개의 값 a, b와 carry input의 값 ci를 입력받아 가산하여, 합 s와 carry out co를 출력한다.

D. Ripple Carry Adder (RCA)

- Ripple Carry Adder란 1bit Full Adder를 연속적으로 엮어서 n bit 가산기를 구현한 것으로 carry가 bit 수만큼 단계적으로 전달된다.
- 4bit Ripple Carry Adder의 회로도



E. 이진수의 표현

- Unsigned 표현

  자주 사용하는 이진수 표현은 Unsigned number로의 표현으로 n-bit의 이진수 일때 숫자를 0부터 2^{n-1}까지 표현 가능하다. 예를 들어 4bit 이진수 인경우 0부터 15까지 표현이 가능하다.

- Sign/Magnitude 표현

  Unsigned에서는 모든 숫자가 수의 크기를 나타내었다면 Sign/Magnitude로 표현한 이진수는 왼쪽 끝 자리를 부호 자리로 두어 0인 경우 양수, 1인 경우 음수이다. 예를 들어 1010_2인 경우 -2이다.

- 2's Complement

  Sign/Magnitude와 마찬가지로 왼쪽 끝자리가 부호 자리이지만 수를 적는 방법이 다르다. 예를 들어 -6을 표현하려고 한다면 1110_2이 아니라 -6의 절대값인 6을 먼저 이진수인 0110_2를 보수를 취해 1001_2로 만들고 1을 더하여 2의 보수를 취해주면 1010_2로 2's complement를 통해 음수 표현이 가능하다. N-bit의 이진수에서 2's Complement가 표현 가능한 범위는 $-(2^{n-1})$부터 $2^{n-1}-1$까지 표현 가능하다.

  - 2's Complement가 4bit 이진수인 경우

4bit 2's Complement	Decimal
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	-8
1001	-7
1010	-6
1011	-5
1100	-4
1101	-3
1110	-2
1111	-1

3. 설계 세부사항

A. XOR GATE

■ XOR GATE의 진리표와 Karnaugh map

INPUT		OUTPUT
a	b	y
0	0	0
0	1	1
1	0	1
1	1	0

b \ a	0	1
0	0	1
1	1	0

진리표를 보고 y에 대한 Karnaugh map을 나타냈으며, $Y = \bar{A}B + \bar{B}A$로 표현할 수 있다.

B. Half Adder

■ Half Adder의 진리표와 Karnaugh map

INPUT		OUTPUT	
a	b	s	co
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Output s에 대한 Karnaugh map은 아래와 같다.

b \ a	0	1
0	0	1
1	1	0

이를 보면 XOR gate 때와 같은 카르노맵을 가지고 있으므로 $\boldsymbol{s = a \oplus b}$로 표현할 수 있다.

Output co에 대한 Karnaugh map은 아래와 같다

b \ a	0	1
0	0	0
1	0	1

따라서 $co = a \cdot b$로 표현할 수 있다.

C. Full Adder

■ Full Adder의 진리표와 Karnaugh map

INPUT			OUTPUT	
a	b	ci	s	co
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Output s에 대한 Karnaugh map은 아래와 같다.

ci \ a b	00	01	11	10
0	0	1	0	1
1	1	0	1	0

이를 정리하면 $s = \bar{a}\bar{b}(ci) + \bar{a}b(\bar{ci}) + ab(ci) + a\bar{b}(\bar{ci})$로 표현할 수 있으며 이는 $s = a \oplus b \oplus c$로도 표현할 수 있다.

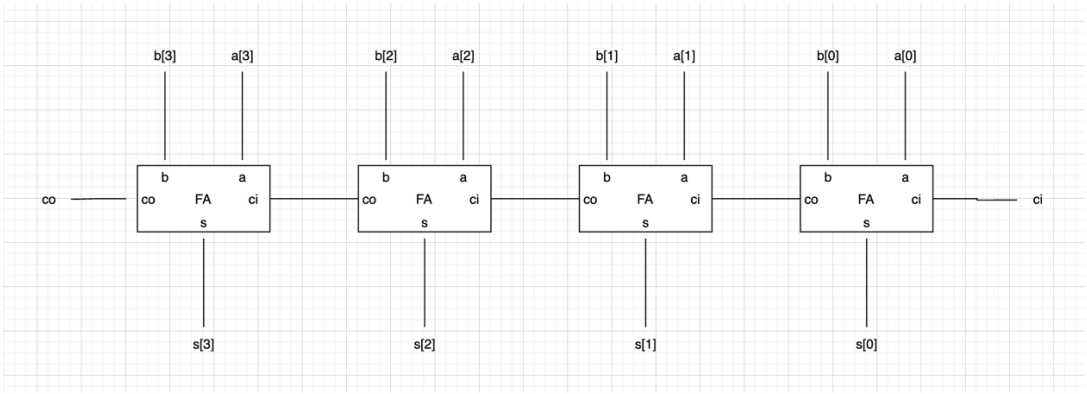
Output co에 대한 Karnaugh map은 아래와 같다.

co \ a b	00	01	11	10
0	0	0	1	0
1	0	1	1	1

따라서 $co = ab + a(ci) + b(ci)$이다.

위를 통합하여 보았을 때 Full Adder은 2개의 Half Adder와 2input or gate를 통해 구현할 수 있다.

D. Ripple Carry Adder

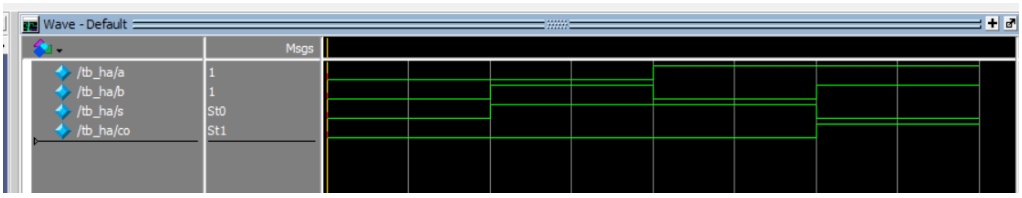


Ripple Carry Adder은 Full Adder를 연속적으로 연결하여 4bit Ripple Carry Adder 구현이 가능하다.

4. 설계 검증 및 실험 결과

A. 시뮬레이션 결과

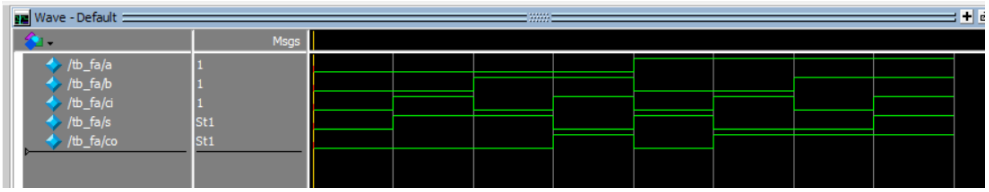
■ Half Adder



Half Adder에 input a, b를 설정해 0,0 / 0,1 / 1,0 / 1,1의 값을 대입하여 output s와 co의 값을 확인하는 testbench를 만들어 시뮬레이션을 돌렸으며 waveform을 분석하면 이전에 Half Adder에 관련해 만들어 둔 진리표와 출력이 같음을 확인할 수 있다.

INPUT		OUTPUT	
a	b	s	co
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

■ Full Adder

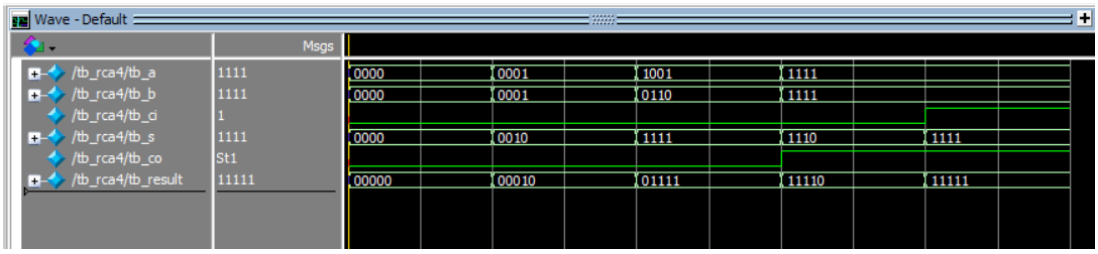


Full Adder에 input a, b, ci를 설정하고 000부터 111까지의 값을 대입하여 output s, co를 확인하는 testbench를 만들어 시뮬레이션을 돌렸으며 waveform을 분석하면 이전에 Full Adder에 관련해 만들어 둔 진리표와 출력이 같음을 확인할 수 있다.

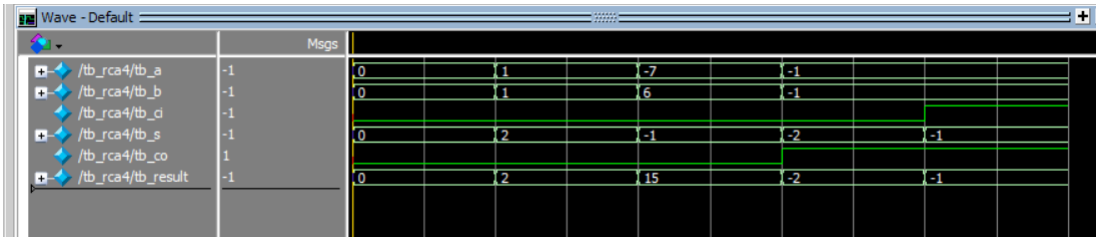
INPUT			OUTPUT	
a	b	ci	s	co
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

■ Ripple Carry Adder

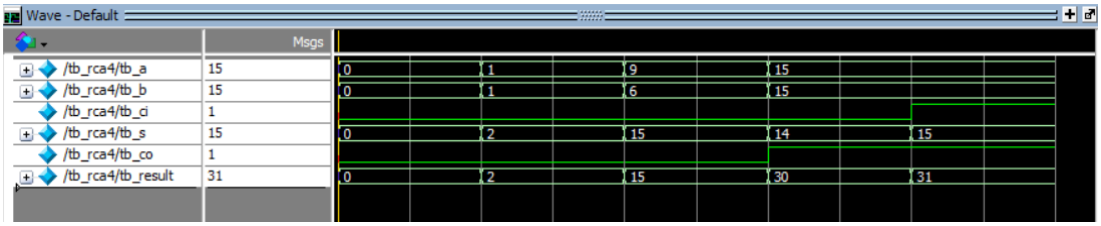
◆ Binary 결과



◆ Decimal 결과



◆ Unsigned 결과



4bit RCA에 임의의 4bit a, 4bit b, ci에 특정 값을 넣어서 output 4bit s, co, 그리고 5bit result를 확인하는 directed verification testbench를 만들어 시뮬레이션 하였다. 경우의 수를 5개 선정하여 기준은 하나의 Full adder에서 co가 다음 Full adder 로 넘어가는 것을 확인할 수 있을 것, 겹치지 않을 때 Ripple Carry Adder의 작동을 확인 할 수 있을 것, 모두가 겹칠 때 Ripple Carry Adder의 작동을 확인 할 수 있을 것, 그리고 처음 ci가 입력되었을 때 Ripple Carry Adder의 작동을 확인 할 수 있을 것이었다.

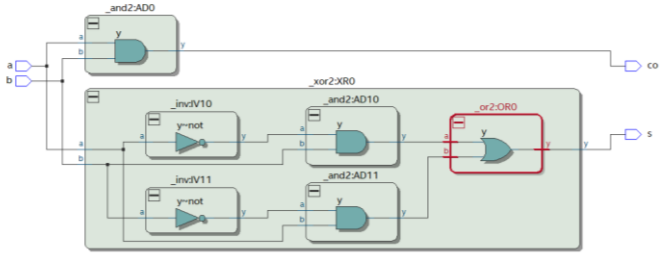
Binary 때와 Unsigned 일 때 결과를 확인하여 표로 정리하면 아래와 같다.

Input			Output		Result
ci	a[3:0]	b[3:0]	co	s[3:0]	co+s[3:0]
0	0000_2 (0)	0000_2 (0)	0	0000_2 (0)	00000_2 (0)
0	0001_2 (1)	0001_2 (1)	0	0010_2 (2)	00010_2 (2)
0	1001_2 (9)	0110_2 (6)	0	1111_2 (15)	01111_2 (15)
0	1111_2 (15)	1111_2 (15)	1	1110_2 (14)	11110_2 (30)
1	1111_2 (15)	1111_2 (15)	1	1111_2 (15)	11110_2 (31)

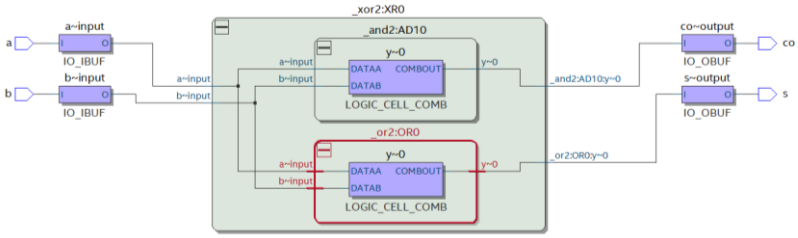
Decimal 결과 같은 경우 예상과 다른 값이 나오는 것을 확인할 수 있는데 이는 Decimal로 Radix를 수정하는 경우 2's Complement를 통해 음수와 양수를 표현하기 때문에 제일 왼쪽에 있는 이진수 자리는 부호를 나타내도록 되기 때문에 Binary 결과와 Unsigned 결과와 다른 결과가 나옴을 위의 waveform을 통해 볼 수 있다. 예를 들어 1111_2은 unsigned에서는 15이지만 decimal에서는 -1이다.

B. 합성(synthesis) 결과

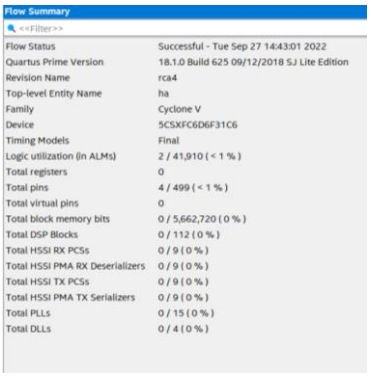
- Half Adder
  - RTL Viewer



  - Technology Map Viewer



  - Flow Summary

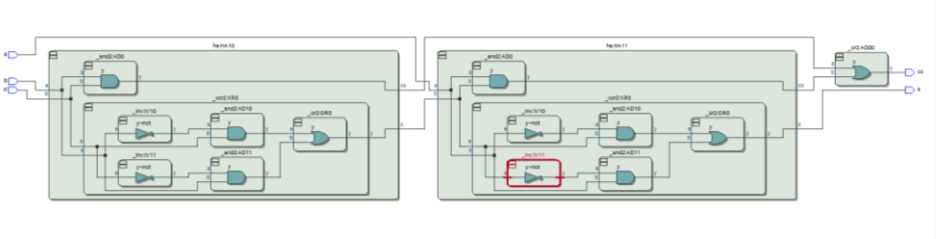


Flow Summary	
Flow Status	Successful - Tue Sep 27 14:43:01 2022
Quartus Prime Version	18.1.0 Build 625 09/12/2018 SJ Lite Edition
Revision Name	rca4
Top-level Entity Name	ha
Family	Cyclone V
Device	5CSXFC6D6F31C6
Timing Models	Final
Logic utilization (in ALMs)	2 / 41,910 (< 1 %)
Total registers	0
Total pins	4 / 499 (< 1 %)
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 %)

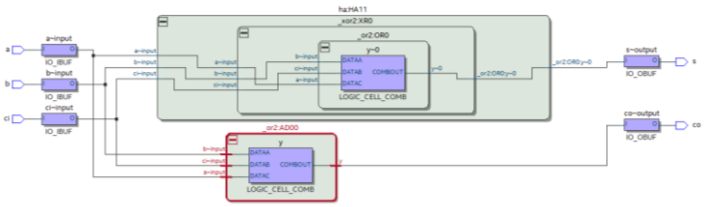
직접 만든 2 input and와 xor gate를 사용하여 Half Adder를 구현하였다. RTL viewer에서 요약된 정보를 확장하여 확인하면 위와 같다. Logic utilization을 통해 logic 개수를 파악하고 Top-level Entity Name도 확인하여 top-level도 올바르게 설정함을 확인하였다.

■ Full Adder

◆ RTL viewer



◆ Technology Map Viewer



◆ Flow Summary

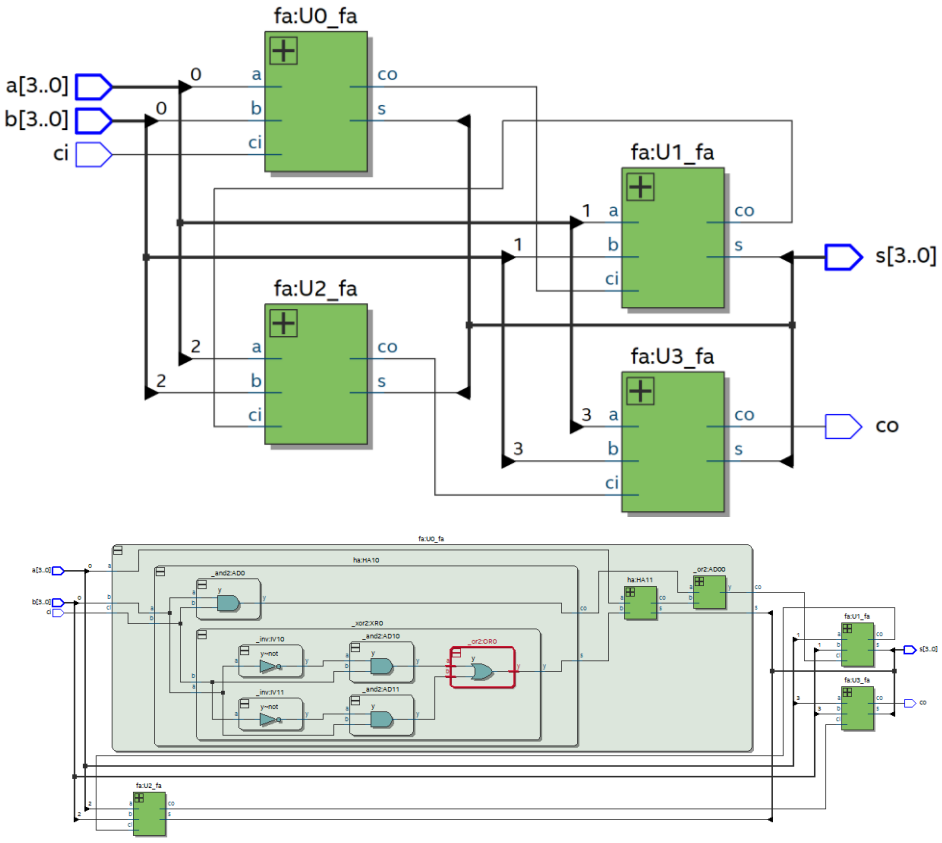
Flow Summary

Flow Status	Successful - Tue Sep 27 14:31:20 2022
Quartus Prime Version	18.1.0 Build 625 09/12/2018 SJ Lite Edition
Revision Name	rca4
Top-level Entity Name	fa
Family	Cyclone V
Device	5CSXFC6D6F31C6
Timing Models	Final
Logic utilization (in ALMs)	2 / 41,910 (< 1 %)
Total registers	0
Total pins	5 / 499 (1 %)
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 %)

2개의 Half Adder와 1개의 2 input or gate를 사용해 총 3개의 유닛을 통해 Full Adder를 구현하였고 RTL viewer에 요약된 정보를 확장하여 확인하면 위와 같다. Logic utilization을 통해 logic 개수를 파악하고 Top-level Entity Name도 확인하여 top-level도 올바르게 설정함을 확인하였다.

■ Ripple Carry Adder

◆ RTL viewer



◆ Flow Summary

Flow Summary	
Flow Status	Successful - Tue Sep 27 14:49:04 2022
Quartus Prime Version	18.1.0 Build 625 09/12/2018 SJ Lite Edition
Revision Name	rca4
Top-level Entity Name	rca4
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 %)

4개의 Full Adder를 연결하여 Ripple Carry Adder를 구현한 것으로 RTL viewer에서 한 개의 Full adder의 요약된 정보를 확장하여 확인하여 Full adder끼리 연결됨을 확인했다. Logic utilization을 통해 logic 개수를 파악하고 Top-level Entity Name도 확인하여 top-level도 올바르게 설정함을 확인하였다.

5. 고찰 및 결론

A. 고찰

이번 실습에서는 gate단부터 4-bit RCA까지 각 모듈 단위로 구현하여 이전에 처음

Verilog를 사용했을 때에 비해 module을 불러오는 등 여러 작업이 수월한 편이었다. 처음에는 익숙하지 않았기 때문에 오히려 1학기 때 컴퓨터공학기초실험 1에서 물리적으로 직접 설계하고 싶다는 생각이 다분했으나 점점 익숙해짐과 동시에 HDL 언어를 통해 4bit RCA를 설계해보면서 각 모듈을 여러 번 사용 가능하고 단선과 같은 문제를 신경쓰지 않아도 된다는 점 마지막으로 검증 하는 것에 있어서 하드웨어를 통해 직접 구현하는 것에 비해 매우 편리하다는 것을 이번 실습을 통해 알게 되었다.

다만 하나의 프로젝트 안에 여러 RTL 뷰어를 한번에 볼 수 있는 것이 아닌 할 때마다 top level을 정하고 tb 또한 수정해서 컴파일 및 시뮬레이션 해야하는 것은 조금 불편한 사항이었다.

B. 결론

여러 gate 그리고 Half adder, Full adder, 4bit RCA를 설계함을 익히고 특히 XOR 게이트에서는 기존에 있는 간단한 연산자를 사용하지 않고 직접 and, or, not을 사용하여 구현할 수 있었다. 또한 Half adder와 Full adder는 Karnaugh map을 통해 알아낸 논리식을 통해 구현하고 이를 통해 4bit RCA 또한 구현하였다.

4bit RCA를 이용하여 32bit RCA를 구현하는 것은 4bit RCA를 구현할 때와 마찬가지로 4bit RCA 8개 각각의 co와 ci를 연결함으로 구현할 수 있다. 4bit RCA에서는 4개의 Full Adder가 사용되었으므로 32bit RCA에서는 32개의 Full Adder가 사용된다. 다만 RCA 구조적 특성상 각 자리 수 계산 후 다음 자리에 carry 값을 넘겨주는 방식 때문에 32bit 계산을 할 경우 4bit에 비해 계산 속도가 느려질 것이다.

6. 참고문헌

1) 공진흥 / 컴퓨터공학기초실험2 week3 / 광운대학교/ 컴퓨터정보공학부
2) 공영호 / 디지털논리회로2 Adder / 광운대학교 / 컴퓨터정보공학ㅂ