컴퓨터 공학 기초 실험2 보고서

실험제목: Ripple carry adder (RCA)

실험일자: 2018년 08월 29일 (수)

제출일자: 2018년 09월 04일 (화)

학 과: 컴퓨터정보공학부

담당교수: 이준환 교수님

실습분반: 수요일 0, 1, 2

학 번: 2015722025

성 명: 정 용 훈

1. 제목 및 목적

A. 제목

Ripple carry adder (RCA)

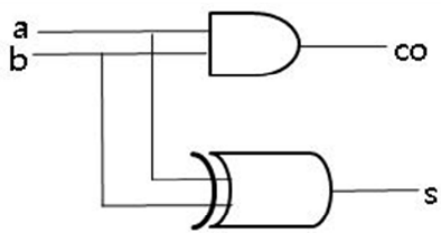
B. 목적

Quartus의 기본적인 기능과 Verilog의 문법을 사용하여 1학기에 배운 기본적인 게이트들을 구성하며, 더 나아가 instance를 통하여 half adder, full adder를 구성하고 이를 이용하여 최종적으로 ripple carry adder를 구성 한다.

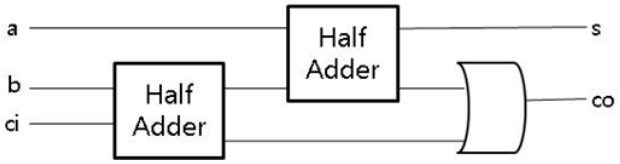
2. 원리(배경지식)

해당 과제를 진행하기 위해서는 기본적으로 unsigned number 와 tow's complement를 알고 있어야한다. unsigned number란 단어 그대로 부호가 없는 수를 의미한다. 컴퓨터에서 수를 나타낼 때 가장 앞에 있는 bit 즉 MSB가 1이면 양수 0이면 음수로 수를 결정하는데 unsigned number는 MSB가 부호의 역할을 하지 않아도 되기 때문에 bit에 비해 나타낼 수 있는 수가 늘어나게 된다는 장점이 있다. 다음은 two's complement이다. two's complement의 수 체계는 앞에서 말한 수체계의 단점을 보완하기 위하여 나온 수체계이다. 앞에서 설명한 수 체계에서는 +0, -0 이렇게 0이 두개가 되는 상황이 생긴다. 이 점을 보완하기 위하여 나온 수 체계이며 현재는 거의 모든 컴퓨터가 two's complement 수 체계를 사용하고 있다. 다음으로는 본격적으로 RCA를 만들기위해 필요한 개념과 Verilog의 기본적인 문법을 알아보겠다. RCA의 기본 구성이 되는 Full adder의 원리와 Full adder를 구성하기 위한 Half adder 그리고 가장 기본적인 게이트들(and, or, xor, 등....)의 진리표와 기능들을 알고 있어야 한다. 세부사항을 설명하기전에 간단하게 RCA와 Full adder, Half adder의 원리를 설명하면 첫째로 Half adder는 input 을 두개 받으며 out put으로는 sum과 carry out이 있다. 둘째로 Full adder는 Half adder와는 다르게 input으로 carry in 값을 하나 더 받을 수 있다. 이는 다른 Full adder에서의 값을 받아 여러 bit의 값을 계산할 수 있도록 설계된 방식이며 이 특징이 RCA를 설계하는데 가장 큰 역할을 한다. 마지막으로 RCA는 원하는 bit수에 따라 Full adder의 개수를 정하여 연결하고 각각의 input에 의하여 output이 결정되며 또한 올림수가 발생하였을 경우 순차적으로 다음 가산기에 input이 전달되는 형식으로 설계가 되어있는 것이 RCA이다. 그리고 우리가 최종적으로 설계를 하기위한 Ripple carry adder가 만들어지기 위하여 Full adder가 필요하며 Full adder가 구성되기 위해서

는 Half adder가 필요하다. 즉 Verilog를 사용하여 각각의 게이트를 구성한 후 instance를 통하여 순차적으로 모듈들을 구성하면 쉽게 해결 할 수 있는 과제이다. 아래의 그림은 Half adder이며 Full adder를 구성하기 위한 모듈이다.

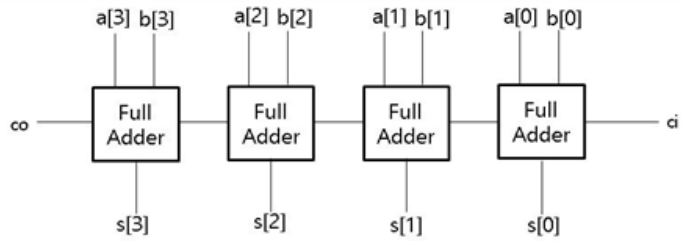


구성한 Half adder를 사용하여 Full adder를 구성할 수 있는데 구성하면 아래와 같은 그림으로 간단하게 볼 수 있다.



이렇게 Full adder까지 구성을 하면 비로소 Ripple carry adder를 구성 할 수 있는 모든 준비가 완료된다.

Full adder를 이용하여 4-bit Ripple carry adder를 구성하면 아래와 같은 symbol로 보여질 수 있다.



위 내용은 RCA의 원리와 구성하는데 있어서 필요한 재료와 구성 순서를 알려주는 내용이었다. 직접적으로 프로그램을 통하여 하드웨어를 설계하기 위해서는 Verilog 문법을 알아야한다. 우리가 쓰기 위한 module 즉 게이트들은 module이라는 명령어를 시작으로 endmodule 이라는 명령어로 끝이 난다 그 안에 우리가 원하는 게이트들의 기능을 설계하면 하드웨어를 설계하며 사용 할 수 있는 module이 생성이 된다. 특히 이번과제는 instance를 익히기 위하여 기본적인 게이트도 스스로 설계하며 설계한 module로 좀 더 높은 level의 module를 설계하는 방법을 제시하고 있다.

3. 설계 세부사항

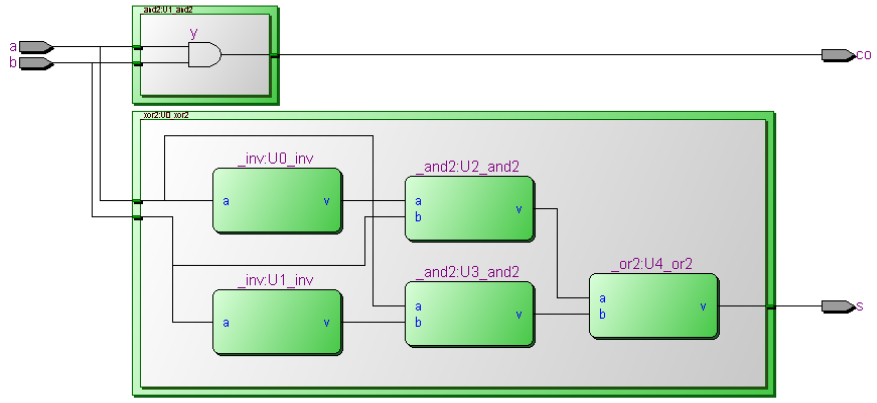
가장 처음 설계한 Half adder, 순서데로 Full adder, 4-bit RCA에 대한 불 식과 진리표를 나타내면 아래와 같이 나타낼 수 있습니다.

a. Half adder

Input		Output	
a	**b**	**s**	**Co**
0	**0**	**0**	**0**
1	**0**	**1**	**0**
0	**1**	**1**	**0**
1	**1**	**0**	**1**

$$Co = ab\ ,\ S=a\oplus b$$

Half adder는 위와 같은 진리표와 식을 갖는다. 하드웨어를 구현 하는 방법으로는 아래와 같은 그림으로 and gate하나 inverter 2개 and gate 2개 or gate 하나로 구현 가능하다. Verilog 로 구현 할 할 때 미리 구현한 gate들의 정보를 가져와 input과 output을 알맞게 연결하면 된다. 하드웨어의 기능으로는 Input으로 정한 두개의 값을 더하여 결과에 따라 s가 결정되며 carry out은 두 값을 정하여 수올림이 발생하면 1이 올라가게 되어있다.

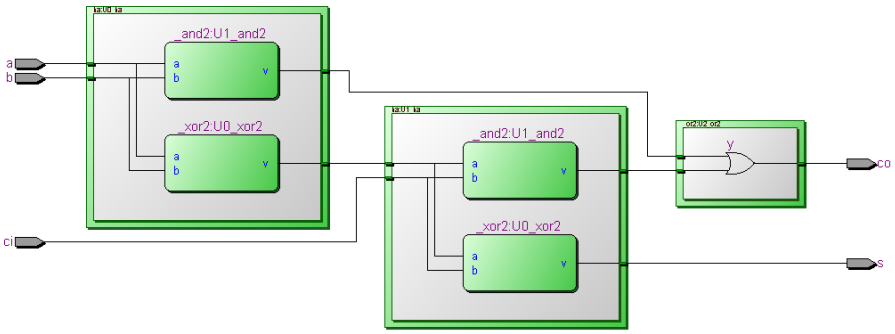


b. Full adder

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**

$$Co = ci\cdot(a\oplus b)+a\cdot b\ ,\ S = a\oplus b\oplus c$$

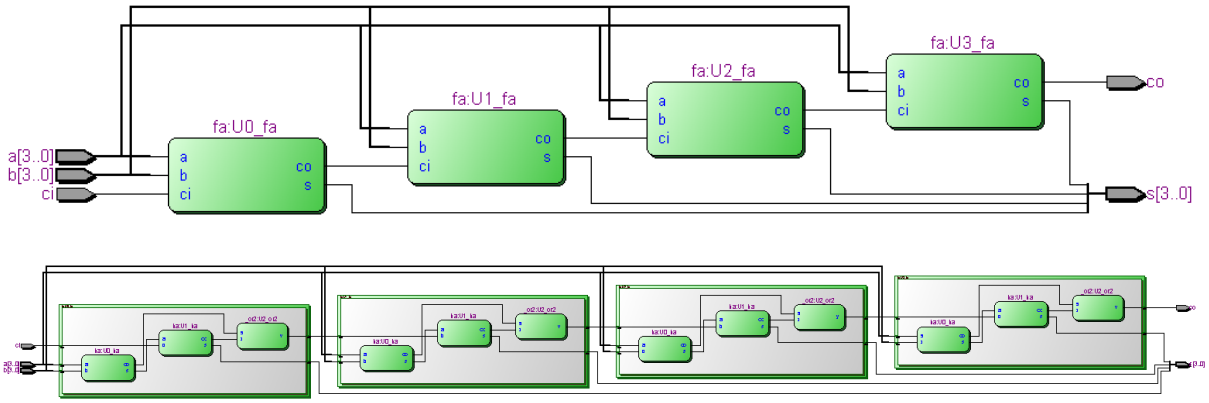
아래 그림은 미리 설계해둔 Half adder와 or gate로 Full adder를 설계한 모습이다. Half adder와는 다른 점으로 carry in을 input으로 하나 더 받으며 이는 여러 비트의 가산기를 붙여 설계하는 RCA를 설계하는데 가장 큰 특징이 된다 계산의 방법은 half adder와 동일 하며 크게 다르지않다.



c. 4-bit Ripple carry adder

우리가 최종적으로 설계하기 위한 4-bit RCA의 모습은 아래와 같다. 여러 비트의 계산을 목적으로 한 adder이므로 입력과 출력 각각 4-bit로 설계하였으며 특징으로는 가산기에서 co이 발생 하면 다음가산기의 ci으로 값이 들어가게 되며 순차적으로 계산하게 되어있다. 마지막 가산기에서 co이 발생하면 output으로 co값이 1 그대로 나오게 설계 되어있다. 따로 구성된 진리표는 없으며 여러 2진수의 덧셈에서 나오는 과정과 결과 값이 CRA의 진리표라고 말해도 상관없다. 모든 상황을 표로 구현하기에는 비효율적이라고 생각되기에 특정 bit를 가지고 표를 아래처럼 구현 해보았다.

a[3:0]	b[3:0]	s[3:0]	co
1010	0011	1101	0
0001	1100	1101	0
1111	1010	1001	1
0011	0011	0110	0



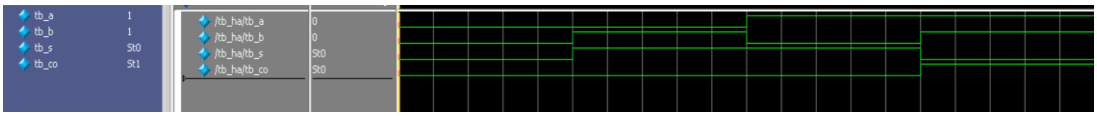
4. 설계 검증 및 실험 결과

A. 시뮬레이션 결과

아래의 이미지는 순서데로 Half adder, Full adder 그리고 4-bit RCA의 waveform을 나타낸 것이다. 각각의 test bench와 결과를 검증하면 아래와 같다.

1. Half adder

Wave form



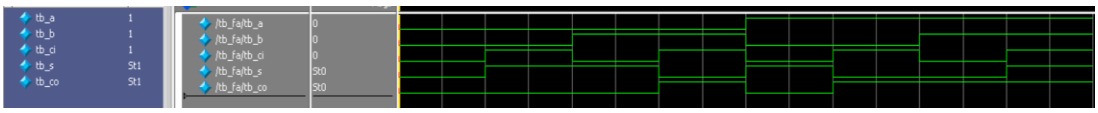
Test bench의 값으로는 아래표와 같은 순서데로 input값을 설정 하였습니다.

a	b
0	0
0	1
1	0
1	1

input으로 는 a와 b의 값을 설정해주었으며 input에 의해 output은 설계한 내용데로 값이 나온다. Input의 값은 설정한 시간이 지나면 바뀔 수 있도록 설계하였으며 그 결과 위와 같은 waveform이 형성이 된다. Waveform을 보면 a와 b가 0일 때 s의 값 과 co의 값이 변화가 없음을 알 수 있고 a가0 b가1 일 때는 s의 값이 1로 증가하는 것을 볼 수 있고 a와 b의 값이 모두 1이면 carry out이 발생하며 co의 값은 1이 되고 s의 값은 다시 0이 되는 상태를 볼 수 있습니다.

2. Full adder

Wave form



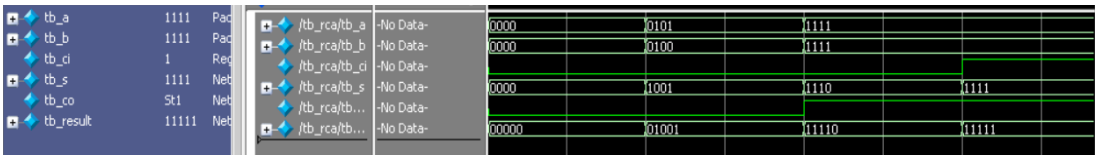
Full adder의 testbench 값은 아래표와 같은 순서데로 input값을 설정 하였습니다.

a	b	ci
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

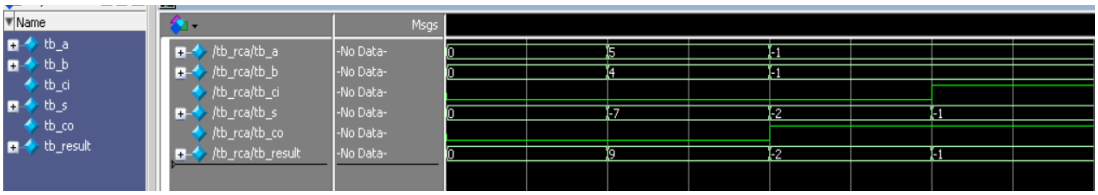
Full adder의 input은 carry in이 추가되어서 나타난다 carry in이라는 input이 있는 이유는 RCA의 설계 구조를 보면 쉽게 알 수 있다. Input과 output에 관하여 설명하자면 초기값은 모두 0이므로 output값도 0인 것을 확인 할 수 있으며 2번째 3번째를 보면 carry in과 b의 값이 1로 올라간 것을 볼 수 있다 그에 따라 s의 값이 1로 증가하는 것을 볼 수 있으며 a, b, ci 중 두개의 input값이 1일 경우는 output값인 carry out이 1로 올라가 있는 것을 확인 할 수 있으며 3개의 input값이 모두 1인 경우는 s의 값과 co의 값이 모두 1로 올라가는 것을 waveform을 보면 확인 할 수 있다.

3. 4-bit RCA

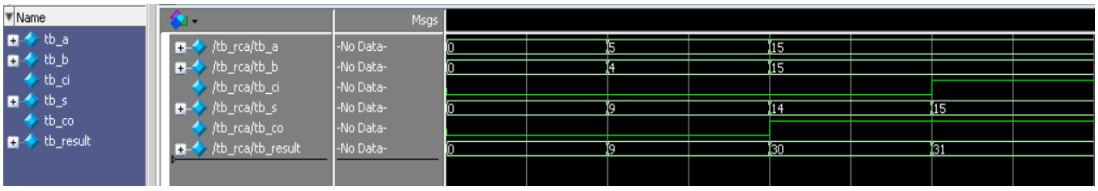
Wave form



Binary



Decimal



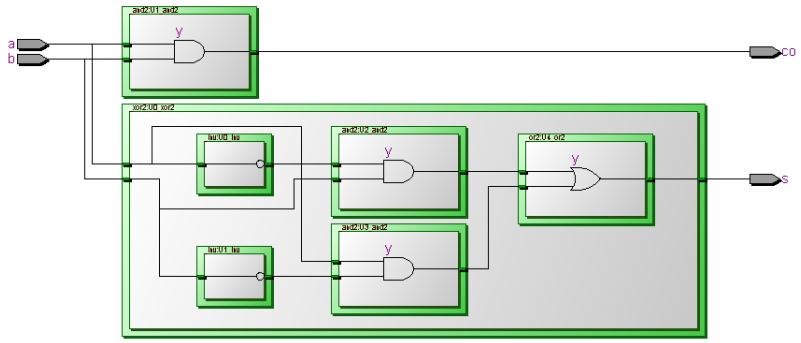
Unsigned

마지막으로 RCA의 test bench이다 위에서 설명된 하드웨어와는 달리 input의 값과 output의 값이 여러 bit로 표현이 되어 Verilog에서 input을 설정하는 방법도 약간의 차이가 있다. 각각의 input에 넣는 값을 16진수로 표기하였으며 예를 들어 5의 값을 입력하면 0101 의 값으로 변환되어 각각의 a input에 0,1,0,1 의 값이 입력된다. 이는 Verilog문법의 특징으로 생각되며 이러한 표기법은 실험할 때 간편할 수 있다. Binary의 결과부터 살펴보면 input a, b를 계산했을 경우 값이 올바르게 나오는 것을 볼 수 있고 carry out이 발생 하였을 경우 co가 1로 올라가는 것을 확인 할 수 있다. Decimal을 확인 하였을 경우 앞서 원리에서 말한 two's complement 와 동일하게 계산이 된 것을 확인 할 수 있고 unsigned는 부호를 결정하는 bit를 사용하지 않고 부호가 양수로 나오므로 단순히 2진수 체계로 계산했을 경우 위와 같은 결과가 나오는 것이 맞다.

B. 합성(synthesis) 결과

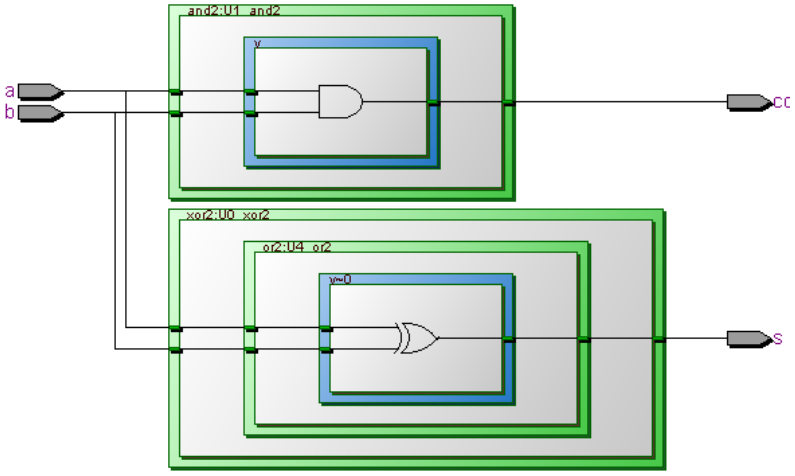
1. Half adder

RTL Viewer



위에는 Half adder가 설계된 것을 가시적으로 보여주는 Viewer이다 and게이트하나와 xor 게이트를 구성하기 위한 게이트들의 집합이 아래에 보인다.

Technology Map Viewer



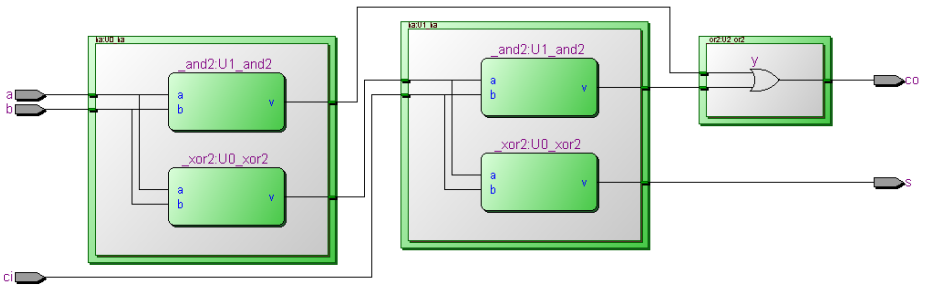
RTL Viewer와 다른 점으로는 xor게이트가 자체가 symbol로 바로 나와있다는 것이 특징이다.

Flow Summary

Flow Summary	
Flow Status	Successful - Sun Sep 02 19:26:22 2018
Quartus II 64-Bit Version	13.0.0 Build 156 04/24/2013 SJ Web Edition
Revision Name	ha
Top-level Entity Name	ha
Family	Cyclone II
Device	EP2C70F896C6
Timing Models	Final
Total logic elements	2 / 68,416 (< 1 %)
Total combinational functions	2 / 68,416 (< 1 %)
Dedicated logic registers	0 / 68,416 (0 %)
Total registers	0
Total pins	4 / 622 (< 1 %)
Total virtual pins	0
Total memory bits	0 / 1,152,000 (0 %)
Embedded Multiplier 9-bit elements	0 / 300 (0 %)
Total PLLs	0 / 4 (0 %)

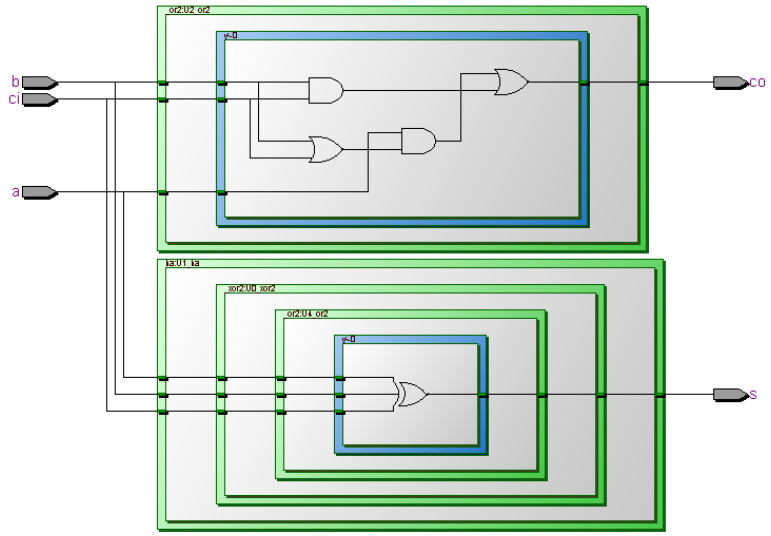
2. Full adder

RTL Viewer



위에서 설명한 Half adder 두개와 or 게이트로 구성되어 있는 것이 Full adder이다.

Technology Map Viewer



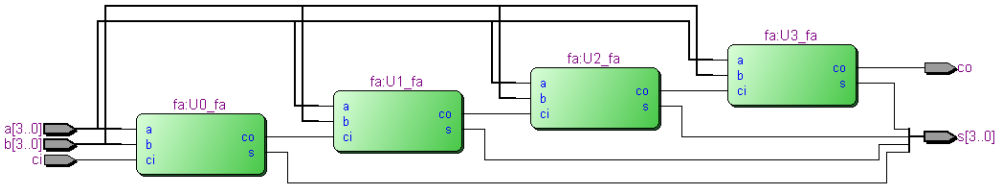
각각의 인풋들이 어떤 과정을 통하여 아웃풋이 결정되는 지를 굉장히 간략하게 나타내 주고있는 Viewer이다.

Flow Summary

Flow Summary	
Flow Status	Successful - Sun Sep 02 19:42:58 2018
Quartus II 64-Bit Version	13.0.0 Build 156 04/24/2013 SJ Web Edition
Revision Name	fa
Top-level Entity Name	fa
Family	Cyclone II
Device	EP2C70F896C6
Timing Models	Final
Total logic elements	2
Total combinational functions	2
Dedicated logic registers	0
Total registers	0
Total pins	5
Total virtual pins	0
Total memory bits	0
Embedded Multiplier 9-bit elements	0
Total PLLs	0

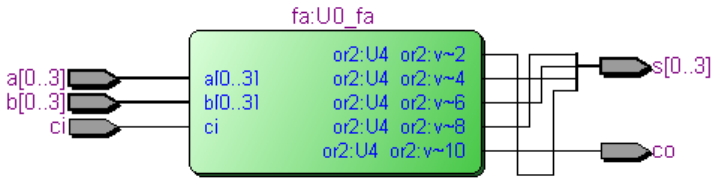
3. 4-bit RCA

RTL Viewer



RCA의 RTL viewer는 Full adder를 4개 연결해 놓은 모습이다 이유는 이번 설계에서 4-bit를 설계하기 위해서 4개를 연결한 모습이다. 좀 더 많은 bit를 계산하고 싶으면 연결 개수를 늘여서 사용하면 된다.

Technology Map Viewer



Flow Summary

Flow Summary	
Flow Status	Successful - Sun Sep 02 19:53:20 2018
Quartus II 64-Bit Version	13.0.0 Build 156 04/24/2013 SJ Web Edition
Revision Name	rca
Top-level Entity Name	rca
Family	Cyclone II
Device	EP2C70F896C6
Timing Models	Final
Total logic elements	6 / 68,416 (< 1 %)
Total combinational functions	6 / 68,416 (< 1 %)
Dedicated logic registers	0 / 68,416 (0 %)
Total registers	0
Total pins	14 / 622 (2 %)
Total virtual pins	0
Total memory bits	0 / 1,152,000 (0 %)
Embedded Multiplier 9-bit elements	0 / 300 (0 %)
Total PLLs	0 / 4 (0 %)

5. 고찰 및 결론

A. 고찰

처음으로 Verilog와 관련된 프로그램과 언어를 사용하게 된 실험이었습니다. RCA의 기능과 원리를 이해하는데 있어서는 1학기에 배웠기에 많은 어려움은 없었지만 이번학기에 새로 배우기 위한 Verilog와 관련된 프로그램 설치부터 시작해서 프로그램을 사용하는데 있어 익숙하지 않았고 어려운 부분이 많았습니다. 우선 낯설었던 점이 프로젝트의 이름과 모듈의 이름이 같아야 한다는 점부터 시작해서 시뮬레이션을 돌리기 위한 셋팅부터 굉장히 많은 시간이 걸렸습니다. 해결 방법으로는 올라온 강의자료를 최대한 적극 사용하였 익숙해질 때까지 반복적으로 읽어본 것이 큰 도움이 되었습니다.

B. 결론

우선 이렇게 하드웨어를 프로그램으로 작성 할 수 있다는 것을 듣기만 해보고 직접 사용한 경험이 없었습니다. 직접 설계 전에 프로그램을 사용하면 좀 더 자세하고 정확하게 하드웨어를 설계 할 수 있을 것 같다는 생각을 하게 되었고 테스트 벤치를 사용하여 원하는 값을 넣어 시뮬레이션을 할 수 있어서 시간과 비용까지 절약 할 수 있다고 생각됩니다. 실험 적인 내용으로는 이번 실험에서는 4bit RCA를 구현하였습니다. 하지만 Full adder를 더 연결하여 원하는 사용자가 원하는 만큼의 bit를 정하여 하드웨어를 구현하여 사용할 수 있다고 생각 됩니다.

6. 참고문헌

David Money Harris / Digital Design and Computer Architecture (Second Edition) / Elsevier Inc / 2013

RCA와 Full adder / http://www.circuitstoday.com/ripple-carry-adder

Adder의 종류와 기능 / https://ko.wikipedia.org/wiki/%EA%B0%80%EC%82%B0%EA%B8%B0