컴퓨터 공학 기초 실험2 보고서

실험제목: Ripple carry adder (RCA)

실험일자: 2023년 09월 18일 (월)

제출일자: 2023년 09월 22일 (금)

학 과: 컴퓨터정보공학부

담당교수: 이준환 교수님

실습분반: 월요일 0, 1, 2

학 번: 2022202065

성 명: 박나림

1. 제목 및 목적

A. 제목

Ripple carry adder (RCA)

B. 목적

RCA 회로의 원리에 대해 공부하고 이를 이해하여 설계할 수 있도록 한다. 단계 별로 만들 수 있도록, 논리 게이트들부터 하여 Half adder를 설계하도록 한다. 이를 이용하여 Full adder를 만들고, 최종적으로 RCA까지 만드는 걸 목적으로 한다.

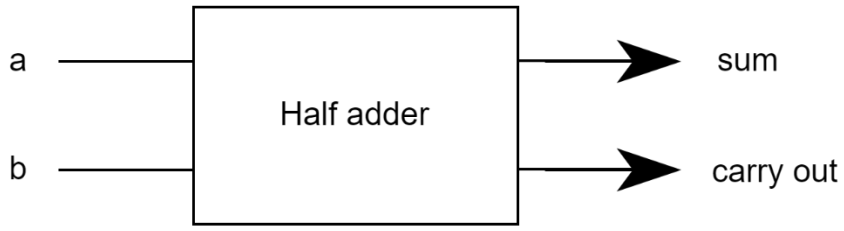
2. 원리(배경지식)

A. 2의 보수 (Two's complement)

주로 기존의 수를 음수로 변환하고자 할 때 이용하는 이진수로, 어떠한 수를 2의 제곱수에서 빼서 나온 결과를 말한다. 컴퓨터에서는 음수를 표현할 방법이 따로 없기 때문에, 부호 절대값 방법으로 하다가 0의 이중 표현이나 덧셈 등 연산하는 데에 불편함이 있어 보수의 방법을 이용하고자 하는 것이다. 그 중에서도 2진법에서 이용되는 2의 보수가 있는데, 2의 보수를 구하고자 할 때는 먼저 주어진 이진수의 모든 자리의 수를 반전시킨 다음에 1을 더하여 구할 수 있다.

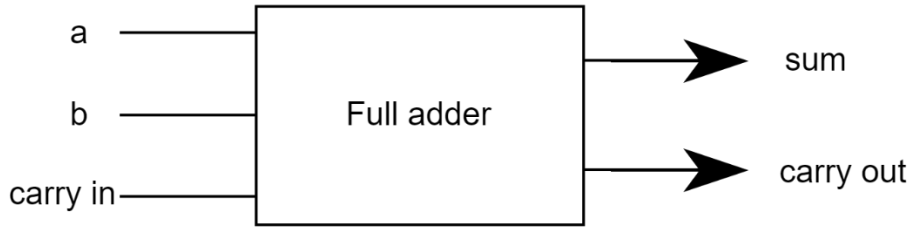
예시로, 6-3을 한다고 하면 0110 – 0011이 연산 되어야 하므로 뒤의 0011을 반전시켜 1100으로 만든 다음 1을 더하여 1101로 만든다. 그렇게 하면 6+(-3)의 형태가 되어 0110+1101=1 0011가 나올 수 있게 된다. 이때 발생한 carry는 제외시킨다.

B. Half adder



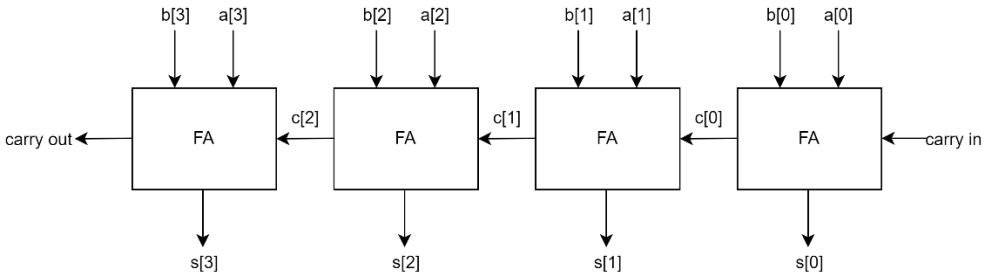
덧셈 연산을 수행하는 논리 회로를 가산기라 하는데, 이 중에서 이진수의 한 자릿수를 연산하고 발생한 carry는 carry out으로 출력하는 가산기를 반가산기 (Half adder)라고 한다. 이진수를 각각 입력으로 더하여 (a+b) 0과 1까지는 sum으로 출력하고 합이 2가 되면 sum이 0이 되고 carry out으로 1을 출력하는 방식이다.

C. Full adder



가산기 중에서, 이진수의 한 자릿수를 연산하고 발생한 carry를 그 다음 자리올림수로 받아서(carry in)과 함께 덧셈 연산을 하는 가산기를 전가산기 (Full adder)라고 한다. 이진수와 carry in을 각각 더하여 (a+b+carry in) 마찬가지로 0과 1까지는 sum으로 출력하고 합이 2가 되면 sum은 0이 되고 carry가 그 다음 자리올림수로 넘어가서 carry in이 되는 방식이다.

D. Ripple carry adder (RCA)



n bit인 두 이진수를 더할 수 있는 가산기로, n만큼 full adder를 연결하여 구현할 수 있다. 각각의 전가산기가 자리올림수로 carry in을 받아서 다음의 carry out으로 넘어가는 형식이 물결치듯 옮겨가는 모습 같다 하여 ripple carry adder라 한다.

위 그림은 4 bit RCA를 구현한 모습이다. 이러한 RCA는 구현이 비교적 간단하며, carry가 발생하지 않을 경우엔 다른 가산기와 속도가 비슷하게 나올 수 있다는 장점이 있다. 하지만 최악의 경우, 각각의 Full adder의 연산 시간을 전부 더한 속도로 나올 수 있기 때문에, 연산 속도가 느리다는 단점이 있다.

3. 설계 세부사항

반가산기를 이용하여 전가산기를 설계하고, 4개의 전가산기들을 연결하여 4-bit RCA를 설계하도록 한다.

A. Half adder

1) Truth Table

input		Output	
a	b	co	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

2) K-map

-co (carry out)

a \ b	0	1
0	0	0
1	0	1

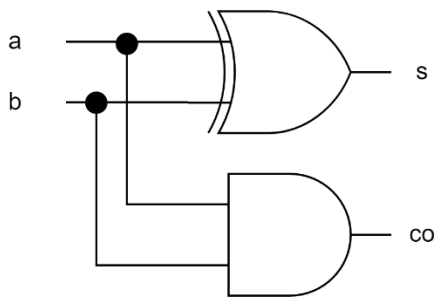
➔ co = ab

-s (sum)

a \ b	0	1
0	0	1
1	1	0

➔ s = ab′ + a′b = a⊕b

3) Circuit



B. Full adder

1) Truth Table

Input			Output	
ci	a	b	co	s
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

2) K-map

- s (sum)

ci \ ab	00	01	11	10
0	0	1	0	1
1	1	0	1	0

->s = a′b′ci + a′bci′ + abci + ab′ci′ = ci(a′b′ + ab) + ci′(a′b + ab′) = ci(a⊕b) + ci′(a⊕b)

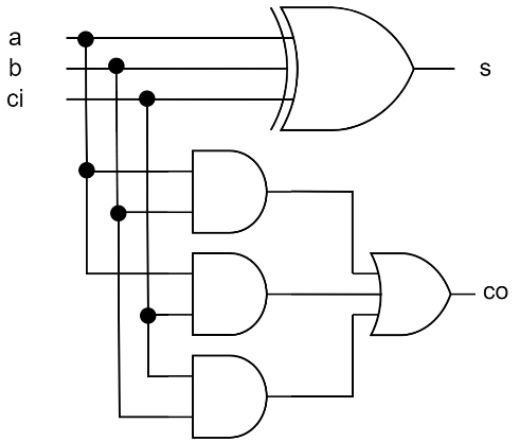
= a ⊕ b ⊕ ci

- co (carry out)

ci \ ab	00	01	11	10
0	0	0	1	0
1	0	1	1	1

➔ co = ab + aci + bci

3) Circuit

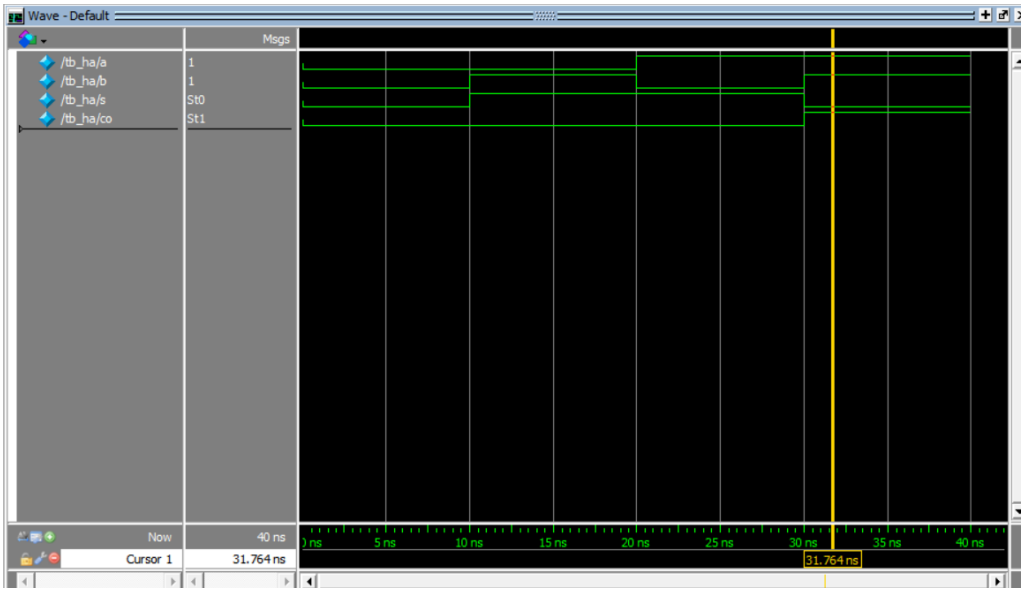


4. 설계 검증 및 실험 결과

A. 시뮬레이션 결과

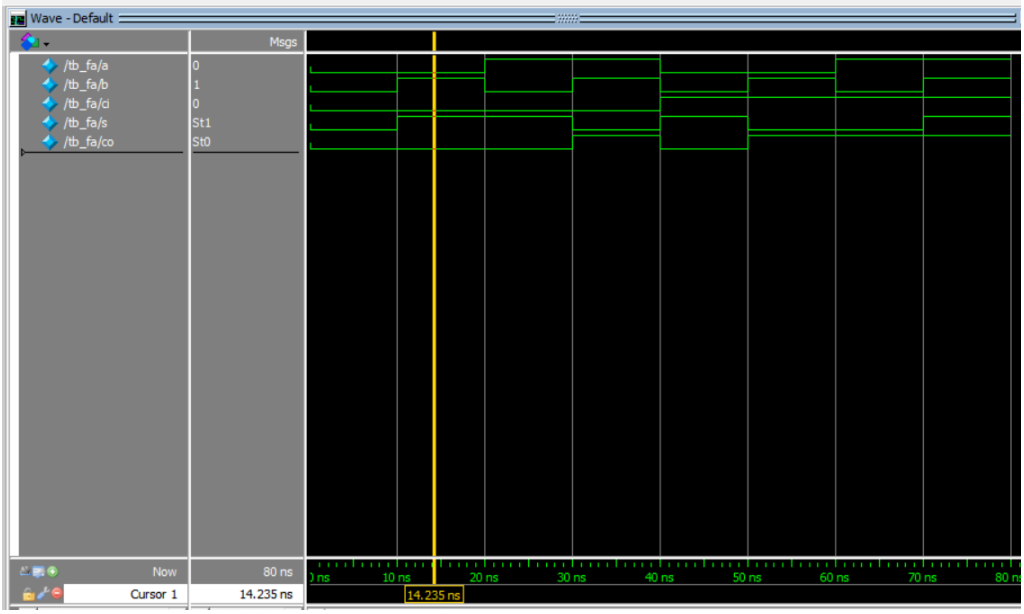
1) Harf Adder

반가산기의 진리표대로 값을 넣었을 때 나온 결과이다. a, b의 값이 전부 1일 때 co가 1로 올라간다.



2) Full Adder

전가산기의 진리표대로 값을 넣었을 때 나온 결과이다. a, b, ci의 값을 입력으로 받아서 carry가 발생할 때만 co가 출력된다.

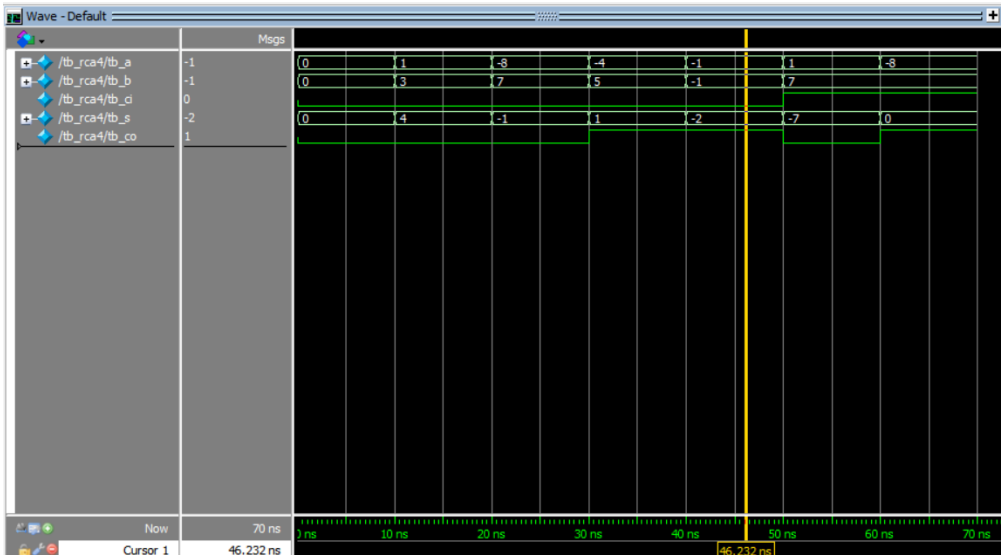


3) 4-bit RCA

4-bit RCA의 입력은 몇 가지만을 선별하여 하였다. 기본적인 덧셈(1~3번째), carry out이 나오는 경우(4, 5번째), carry in까지 1로 한 경우(6, 7번째)로 나누어서 검증을 진행하였다. carry out이 나오는 경우에는 co로 1이 출력되는 것과, ci까지 1로 하면 총 3개의 input값을 더하여 출력하는 결과를 알 수 있다.

-radix를 decimal로 할 때

decimal로 변수 값들을 설정하여 보면 다음과 같다. ci가 0일 동안에는 예상한 덧셈 결과가 나오는 걸 알 수 있다. ci가 1이 되어 총 3개로 덧셈을 진행할 때에는 다른 결과 값이 나온다.



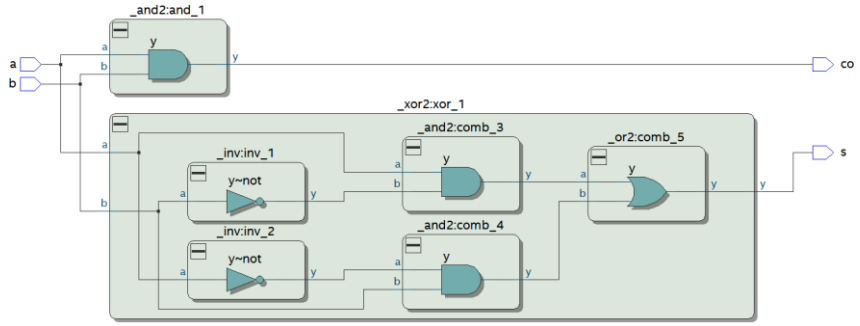
-radix를 unsigned로 할 때

unsigned로 변수 값들을 설정하여 보면 carry out이 나오기 전까지는 예상한 결과 값이 나오는 걸 볼 수 있다. 범위를 벗어날 시 co가 출력되면서 s값이 다르게 나온다. 또한 ci가 1이더라도 carry out이 나오지 않으면 3개를 더하여 정상적으로 s값이 나온다.

B. 합성(synthesis) 결과

1) Harf Adder

-RTL View



반가산기의 회로도로, AND gate와 XOR gate로 구성된 것을 볼 수 있다. XOR gate는 inverter 2개와 and 2개, or 1개로 구성하여 설계하였다.

-flow summary

Flow Summary

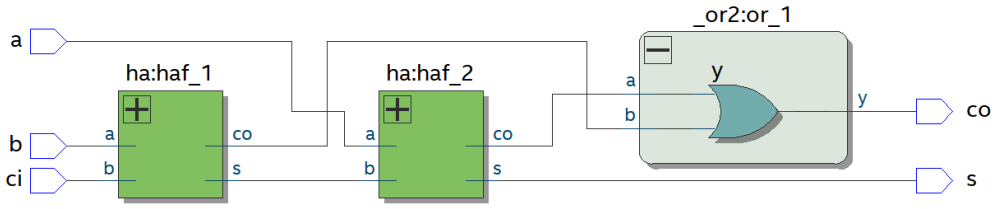
<<Filter>>

Flow Status	Successful - Mon Sep 18 14:05:30 2023
Quartus Prime Version	18.1.0 Build 625 09/12/2018 SJ Lite Edition
Revision Name	ha
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 %)

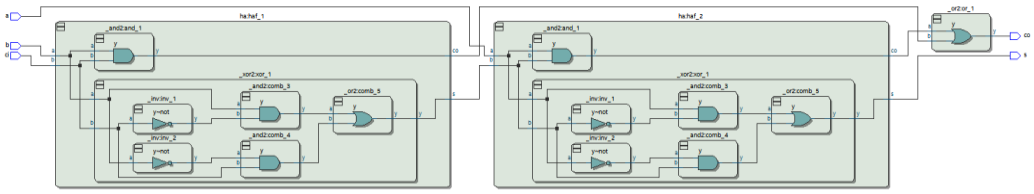
ha.v 파일을 컴파일했을 때 성공적으로 된 모습이다.

2) Full Adder

-RTL View



전가산기는 반가산기 2개와 or gate 1개로 설계되었으며, 아래는 반가산기의 내부 구조까지 확인한 것이다.



-flow summary

Flow Summary

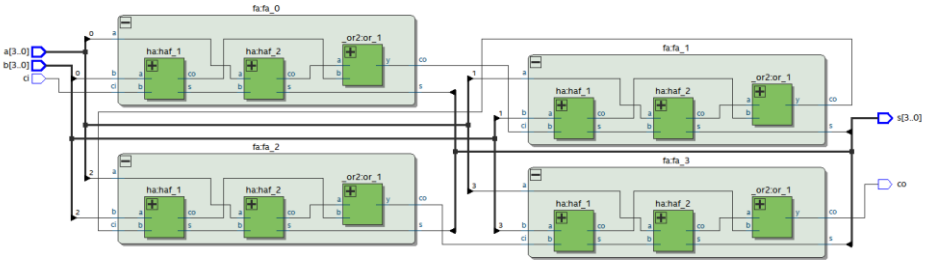
<<Filter>>

Flow Status	Successful - Mon Sep 18 13:38:27 2023
Quartus Prime Version	18.1.0 Build 625 09/12/2018 SJ Lite Edition
Revision Name	fa
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 %)

fa.v 파일을 컴파일했을 때 성공적으로 된 모습이다.

3) 4-bit RCA

-RTL view



4-bit RCA는 전가산기 4개로 연결하였으며, 전가산기 각각은 반가산기와 or gate로 구성되었음을 보여준다.

-flow summary

Flow Summary

<<Filter>>

Flow Status	Successful - Mon Sep 18 15:12:48 2023
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 %)

rca4.v 파일을 컴파일했을 때 성공적으로 된 모습이다.

5. 고찰 및 결론

A. 고찰

각각의 파일들을 컴파일했을 때까지는 무사히 성공했으나, waveform을 보기 위하여 시뮬레이션을 돌렸을 때 'Error loading design'으로 계속 오류가 뜨면서 화면이 출력되지 않는 문제점이 발생하였다. 이에 파일 경로를 확인해보고, 앱을 다시 설치하기도 하였으나 해결되지 않았다. 그래서 컴파일을 돌렸을 때 발생하는 경고문들을 다시 보다가 gates.v 파일에서 코드 오류가 나왔다는 점을 발견하였다. 확인해보니, _xor2 module에서 _inv와 _and module을 instance할 때 새로운 이름을 안 붙였었다. 제대로 수정하고 다시 시뮬레이션을 돌리니 무사히 waveform이 출력되었다.

B. 결론

RCA 회로를 설계하면서 전가산기, 반가산기가 모두 필요하다는 점과 이들은 각각 여러 게이트들로 이루어짐을 알았다. 또한 몇 비트인지에 따라 전가산기 개수가 결정되므로, 이를 통해 4-bit RCA말고도 더 큰 bit의 RCA까지 설계할 수 있음을 깨달았다. 4-bit RCA를 설계한 rca4.v파일을 이용하여 rca32 프로젝트를 생성하여 rca4 모듈을 인스턴스화 하여 8번 불러오면 32-bit RCA를 만들 수 있을 것이다. 이에 테스트 벤치에서도 tb_rca32.v를 만들어서 그에 맞게 적절한 값으로 테스트를 수행하면 결과까지 볼 수 있을 것이다.

6. 참고문헌

David Money Harris and Sarah L. Harris / Digital Design and Computer Architecture / Elsevier / 2007