컴퓨터 공학 기초 실험2 보고서

실험제목: Ripple Carry Adder

실험일자: 2023년 09월 18일 (월)

제출일자: 2023년 09월 24일 (일)

학 과: 컴퓨터정보공학부

담당교수: 이준환 교수님

실습분반: 월요일 0, 1, 2

학 번: 2020202031

성 명: 김재현

1. 제목 및 목적

A. 제목

Ripple Carry Adder

B. 목적

여러 비트의 덧셈을 수행하기 위해 수행한다. 컴퓨터 내에서 이진법의 계산을 파악하고 overflow 상황도 잘 파악할 수 있도록 ripple carry adder를 구성하는 능력을 키울 수 있도록 한다.

2. 원리(배경지식)

1) two's complement

2의 보수는 먼저 1의 보수를 취한 후에 +1을 해주는 과정을 취하는 것입니다. 예를 들면 $8_{10} = 0100_2$에 1의 보수를 취하면 1011_2이고 +1을 하면 1100_2이므로 $1100_2 = -8_{10}$임을 알 수 있습니다.

2) Inverter

input	output
a	y
0	1
1	0

3) AND gate

input		output
a	b	y
0	0	0
0	1	0
1	0	0
1	1	1

4) OR gate

input		output
a	b	y
0	0	0

0	1	1
1	0	1
1	1	1

5) NAND gate

input		output
a	b	y
0	0	1
0	1	1
1	0	1
1	1	0

6) XOR gate

input		output
a	b	y
0	0	0
0	1	1
1	0	1
1	1	0

7) Half Adder

input		output	
a	b	co	s
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

8) Full Adder

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

3. 설계 세부사항

Half Adder

a \ b	0	1
0	0	0
1	0	1

a \ b	0	1
0	0	1
1	1	0

$$co = ab \qquad s = a'b + ab' = a \oplus b$$

Half Adder의 경우 Karnaugh Map으로 Boolean Equation을 구한 결과 co는 and gate로, s는 xor gate로 구현할 수 있음을 알 수 있다.

Full Adder

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

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

$$co = ab + bci + cia$$

$$s = a'bci' + ab'ci' + a'b'ci + abci$$

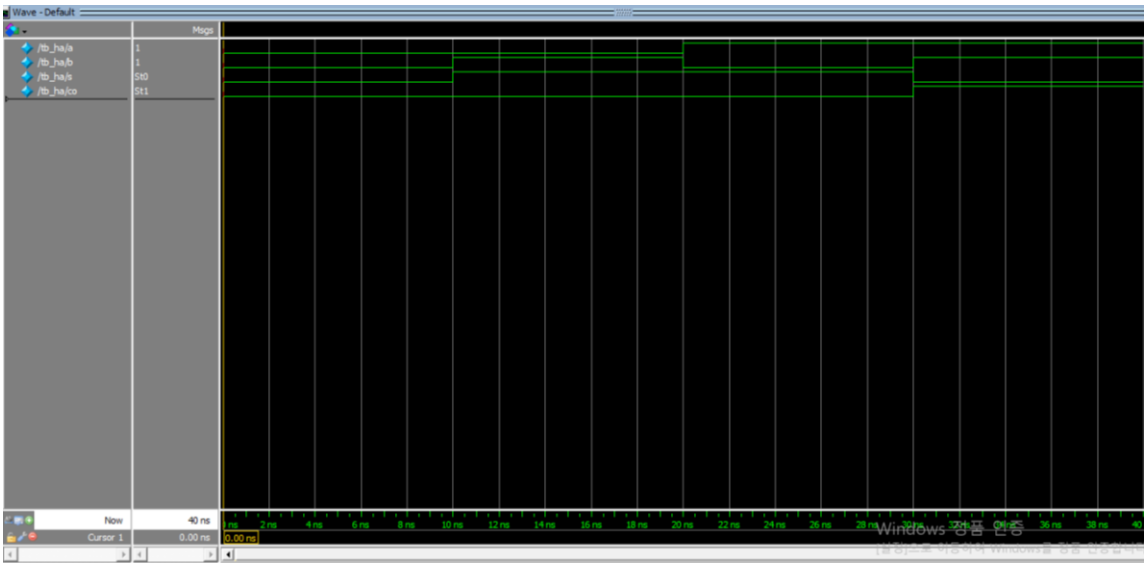
$$s = a \oplus b \oplus c$$

Full Adder의 경우 Karnaugh Map으로 구한 Boolean Equation은 Half Adder 2개와 or gate 1개로 구현할 수 있다. 먼저 input 값 b, ci를 ha1에서 입력 받고, 그에 출력값 sm과 input 값 a를 ha2에서 입력 받는다. ha2에서 도출된 출력 값이 Full Adder의 s값이고, ha1에서 출력된 c1과 ha2에서 출력된 c2를 or gate에 입력해서 도출된 출력 값이 Full Adder의 출력값 co가 된다.

4. 설계 검증 및 실험 결과

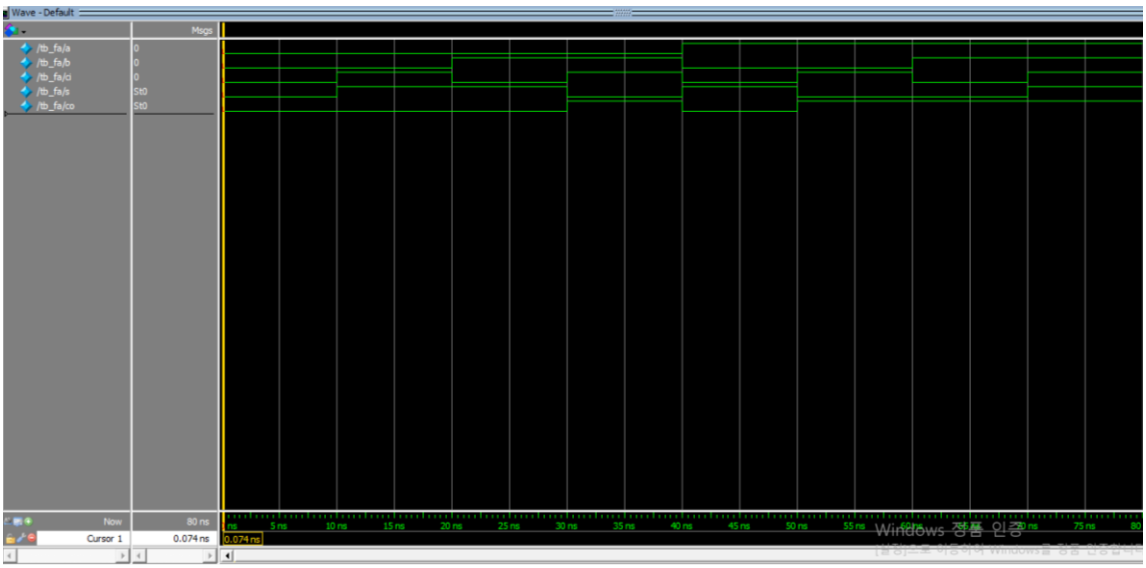
A. 시뮬레이션 결과

Half Adder



ha는 입력 값이 2개이므로 00부터 11까지 총 네 가지의 경우가 존재하므로 input a, b에 네 가지 경우 모두를 대입하여 output s, co가 정상적으로 출력되는 것을 확인할 수 있습니다.

Full Adder



fa는 입력 값이 3개이므로 000부터 111까지 총 여덟 가지의 경우가 존재하므로 input a, b, ci에 여덟 가지 경우 모두를 대입하여 output s, co가 정상적으로 출력되는 것을 확인할 수 있습니다.

Ripple Carry Adder

testbench scenario

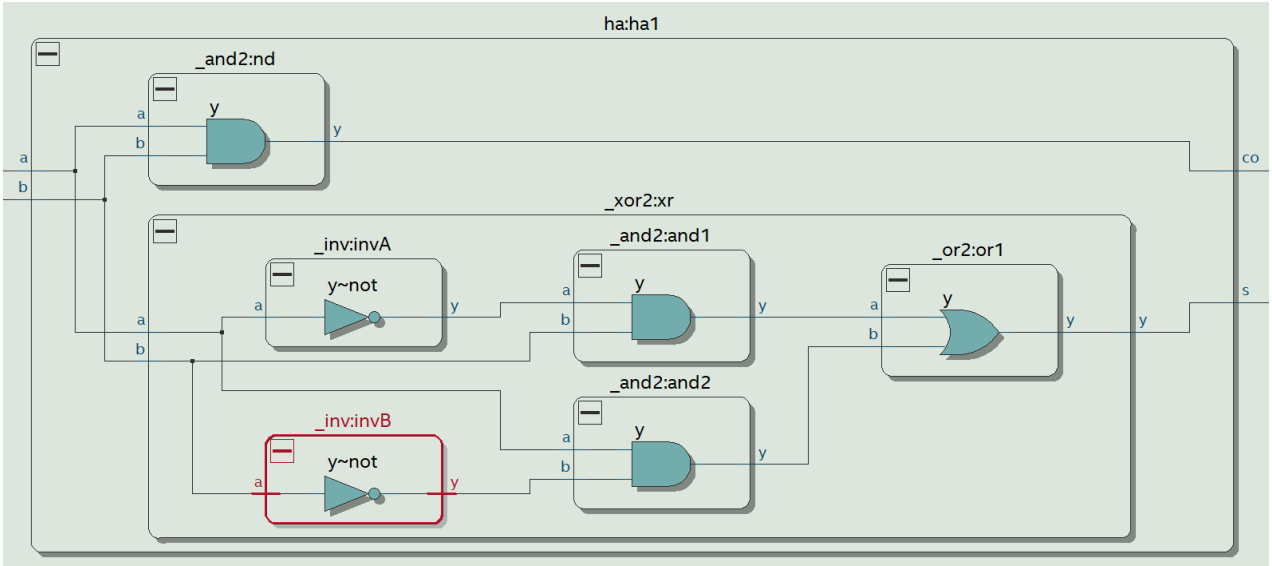
1) 1~6까지의 testbench는 carry out이 발생하지 않는 선에서 각 비트들의 덧셈이 잘 수행되는지 확인하기 위함입니다. 입력 값 a, b, ci에 따라 덧셈이 잘 수행됨을 확인할 수 있었습니다.
2) 7~8 testbench는 carry out이 잘 발생하는지 확인하기 위함입니다. 입력 값 a, b, ci의 덧셈 결과 overflow가 발생한다면 carry out이 발생하는 것이므로 co가 1이 출력되는 것을 확인할 수 있었습니다.

-decimal

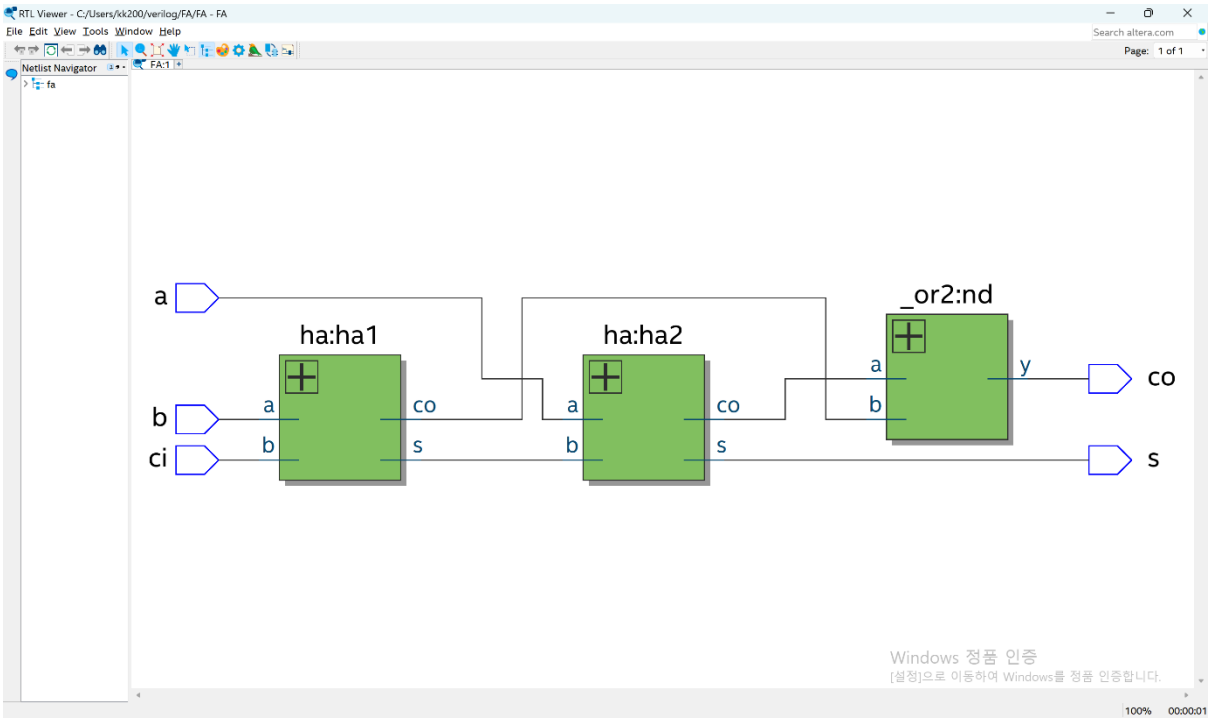
-unsigned

decimal과 unsigned의 출력 값을 (decimal, unsigned)로 나타내면 (0, 0), (1, 1), (3, 3), (-1, 15)이 출력됨을 알 수 있다. unsigned의 15가 decimal의 -1이 되는 이유는 decimal은 -8~7까지의 범위를, unsigned는 0~15까지의 범위를 가지기 때문이다. unsigned의 15는 decimal의 7에서 8만큼 overflow가 일어난 것이기 때문에 -1이 된다고 볼 수 있다. 따라서 구현한 4-bit RCA가 decimal, unsigned 모두에서 잘 작동한다고 할 수 있다.

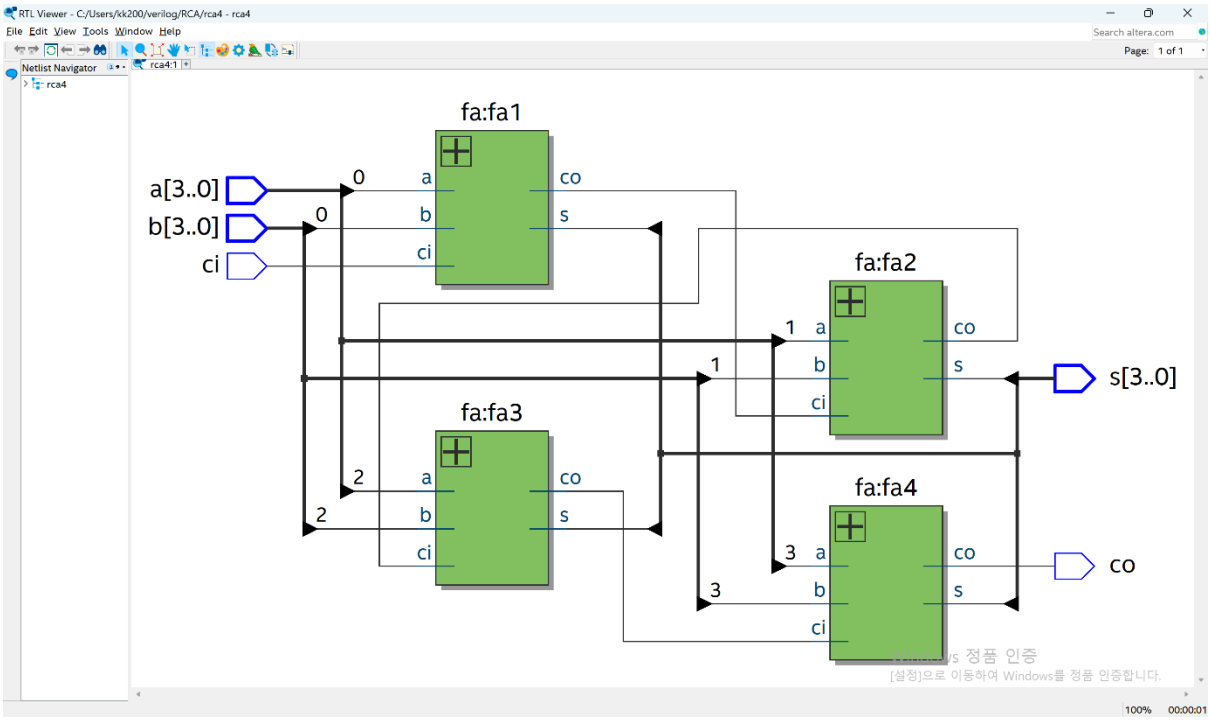
B. 합성(synthesis) 결과



Half Adder는 inverter 2개 and gate 2개, or gate 1개를 사용하는 xor gate와 and gate로 구현했습니다.



Full Adder는 Half Adder 2개와 or gate 1개를 사용하여 구현했습니다.



Ripple Carry Adder는 Full Adder 4개를 사용하여 각 비트를 덧셈 연산하여 s로 출력하도록 설계했으며, 마지막 FA4에서 나온 carryout을 co로 출력하도록 구현했습니다.

Flow Summary

<<Filter>>

Flow Status	Successful - Fri Sep 22 15:41:00 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)	N/A
Total registers	0
Total pins	4
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

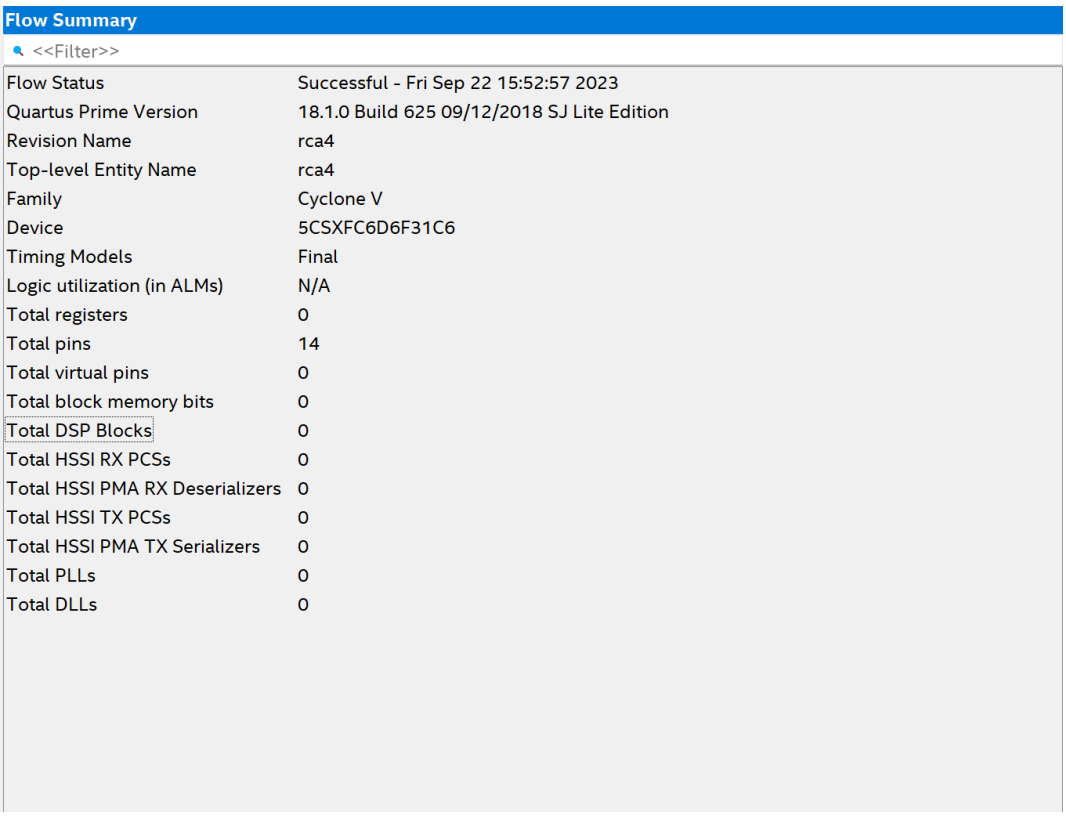
half adder의 flow summary 결과화면이다. 정해진 family와 device를 사용하였고, a, b, co, s 모두 각각 1bit씩 사용하였으므로 Total pins의 값이 4가 나오는 것을 확인할 수 있다.

Flow Summary

<<Filter>>

Flow Status	Successful - Fri Sep 22 15:48:07 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)	N/A
Total registers	0
Total pins	5
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

full adder의 flow summary의 결과화면이다. 정해진 대로 family와 device를 설정하였고, input a, b, ci와 output s, co 모두 각각 1bit씩 할당하여 Total pins의 값이 5가 나오는 것을 확인할 수 있다.



Flow Summary

<<Filter>>

Flow Status	Successful - Fri Sep 22 15:52:57 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)	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

ripple carry adder의 flow summary 화면이다. 정해준 family와 device를 사용하였고, a, b, s는 4bit, ci와 co는 1bit씩 할당했으므로 Total pins 값이 14가 나오는 것을 확인할 수 있다.

5. 고찰 및 결론

A. 고찰

RCA는 1학기 디지털논리회로1 수업에서 이미 한번 배운 적이 있는 내용인데, 회로 구현이 아닌 베릴로그 코딩을 통해 실습을 해보면서, 다시 한번 개념을 짚고 넘어갈 수 있어서 좋았습니다.

베릴로그 코딩할 때, 각 포트들을 어디에 연결해야 할지 머리 속으로 계산하려고 하니 실수가 잦았는데, 수업자료를 참고하거나, 그림을 그려 놓고 하니 실수가 확연히 줄어드는 것을 체감했습니다.

B. 결론

베릴로그 코딩을 해보면서 full adder의 Boolean equation이 half adder를 사용하여 구현할 수 있다는 사실을 다시 한 번 확인할 수 있었다.

4-bit RCA를 사용하여 32-bit RCA를 구현하려면 4-bit RCA를 8개 이어 붙이면 된다. 4-bit RCA는 carry in을 입력 받아 carry out을 출력하는데 8개의 4-bit RCA가 연결돼 있으므로 계속해서 이전 RCA의 carry out이 다음 RCA의 carry in으로 들어가고 결국 마지막 RCA가 출력하는 carry out이 32-bit RCA의 carry out이 되며, 각 비트의 덧셈이 32bit 데이터의 덧셈결과가 된다.

6. 참고문헌

이준환/디지털논리회로2/광운대학교(컴퓨터정보공학부)/2023

이준환/컴퓨터공학기초실험2/광운대학교(컴퓨터정보공학부)/2023