컴퓨터 공학 기초 실험2 보고서

실험제목: Ripple Carry Adder

실험일자: 2021년 09월 20일 (월)

제출일자: 2021년 09월 24일 (금)

학 과: 컴퓨터정보공학부

담당교수: 공진흥 교수님

실습분반: 월요일 0, 1, 2

학 번: 2018202046

성 명: 이준휘

1. 제목 및 목적

A. 제목

2-to-1 MUX

B. 목적

해당 수업을 통해 half adder와 full adder의 동작 방식을 이해한다. 이를 바탕으로 RCA를 제작할 수 있는 능력을 기른다. Directed verification 방식을 통해 4bit RCA를 검증할 수 있다.

2. 원리(배경지식)

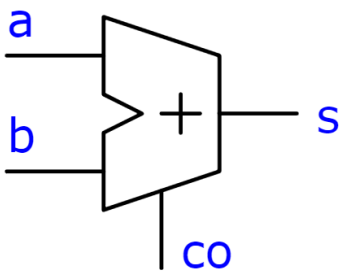
i. Half Adder

half adder는 2개의 1-bit, a와 b를 더할 수 있는 가산기다. a와 b를 더하여 해당 자리의 결과인 s와 올림수인 co를 출력한다.

s와 co의 논리식은 아래와 같다.

$$s = a \oplus b \qquad co = a * b$$

아래 그림은 Half Adder의 symbol이다.



ii. Full Adder

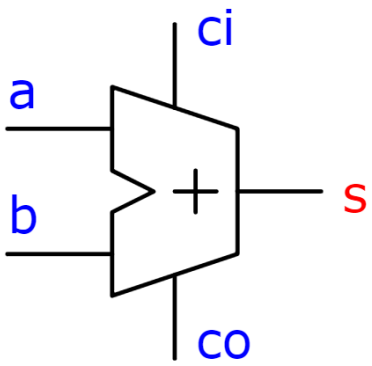
Full adder는 3개의 1-bit, a, b, ci(carry in)를 더할 수 있는 가산기다. a와 b를 더하여 해당 자리의 결과인 s와 올림수인 co를 출력한다.

s와 co의 논리식은 아래와 같다.

$$s = a \oplus b \oplus ci \qquad co = a * b + b * ci + ci * a$$

Full adder는 half adder 2개와 2 input OR Gate를 이용하여 구성할 수 있다.

아래 그림은 Half Adder의 symbol이다.

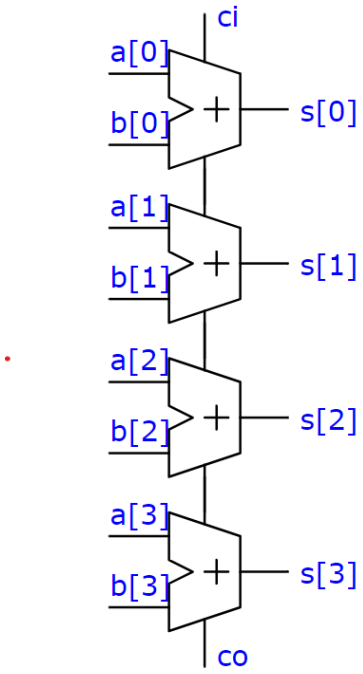


iii. RCA(Ripple Carry Adder)

RCA란 여려 bit를 가지는 두 수를 더하기 위한 가산기로, 더하고자 하는 수의 bit 수만큼 full adder를 연결함으로써 구성된다.

RCA를 이용하여 만약 2개의 sign number를 받을 경우 해당 값을 감산하는 것도 가능하다. 감산을 하기 위해서 하나의 sign number를 invertor를 통해 반전시키고 LSB의 ci에 1을 넣음으로써 2's Compliment를 더하게 한다. 이를 통하여 뺄셈을 구현할 수 있다.

아래의 그림은 4-bit RCA를 나타낸 것이다.



3. 설계 세부사항

i. half adder

half adder의 s를 Karnaugh Map을 통해 확인하면 다음과 같다

a \ b	0	1
0	0	1
1	1	0

위의 Karnaugh Map을 식으로 바꾸면 $s = a * \bar{b} + \bar{a} * b$로 표현할 수 있고, 이 식은 $s = a \oplus b$로 정리할 수 있다.

ci는 Karnaugh Map을 통해 확인하면 다음과 같다

a \ b	0	1
0	0	0
1	0	1

위의 Karnaugh Map을 식으로 바꾸면 $s = a * b$로 표현할 수 있다.

위의 두 식을 통해 ha module을 설계하면 다음과 같다.
s를 output으로 받고, a, b를 input으로 받는 _xor2 module을 통해 XOR Gate를 구현한다.
ci를 output으로 받고, a, b를 input으로 받는 _and2 module을 통해 AND Gate를 구현한다.

ii. full adder

full adder를 해당 과제에서는 half adder 2개와 OR Gate를 사용하여 구현한다.

output sm(wire)와 c1을 가지고, input b와 ci를 가진 half adder를 구현한다. 여기서 sm은 $b \oplus ci$의 값을 가지고 c1은 $b * ci$의 값을 가지게 된다.
그 후 s와 c2를 output으로 가지고, a와 sm을 input으로 가지는 half adder를 하나 더 연결한다. 이 과정에서 s는 a $\oplus b \oplus ci$의 값을 가지게 되고, c2는 $a * (b \oplus ci)$의 값을 가진다.
마지막으로 co를 output으로 가지고, c1과 c2를 input으로 가지는 _or2 module을 연결하여 co = a $* (b \oplus ci) + b * ci = a * b + b * ci + ci * a$를 성립할 수 있도록 한다.

iii. 4-bit RCA

4-bit RCA module에서는 4개의 fa module을 사용한다.

첫 번째 fa module에는 output s[0], c[0](wire)을 연결하고, input a[0], b[0], ci를 연결한다.

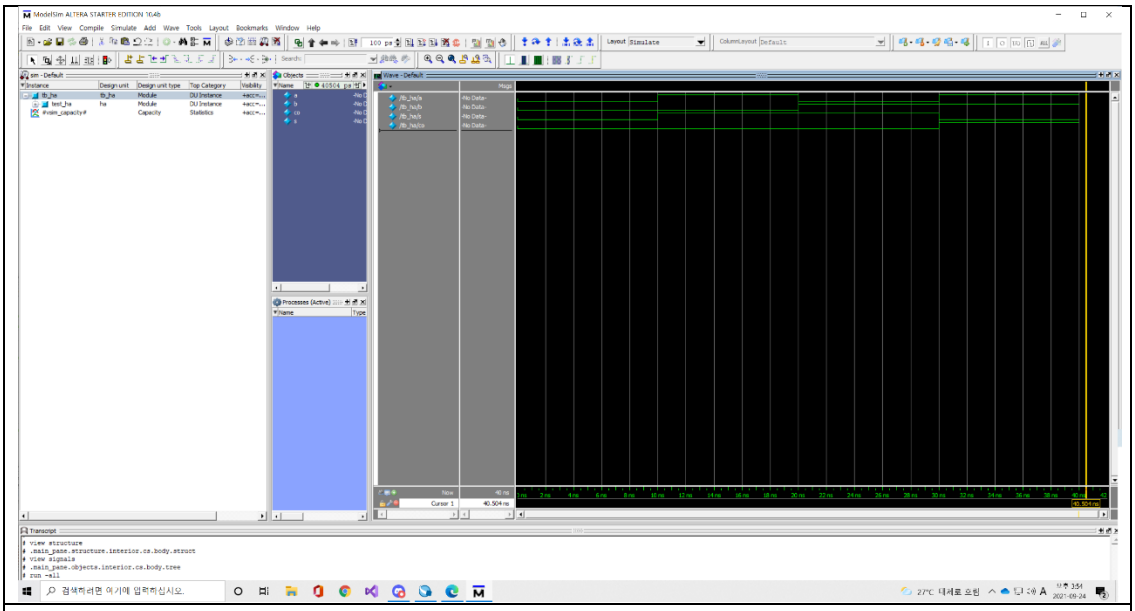
두 번째 fa module에는 output s[1], c[1](wire)을 연결하고, input a[1], b[1]와 이전 fa module에서 넘어온 carry인 c[0]를 연결한다.

세 번째 fa module에는 output s[2], c[2](wire)을 연결하고, input a[2], b[2]와 이전 fa module에서 넘어온 carry인 c[1]를 연결한다.

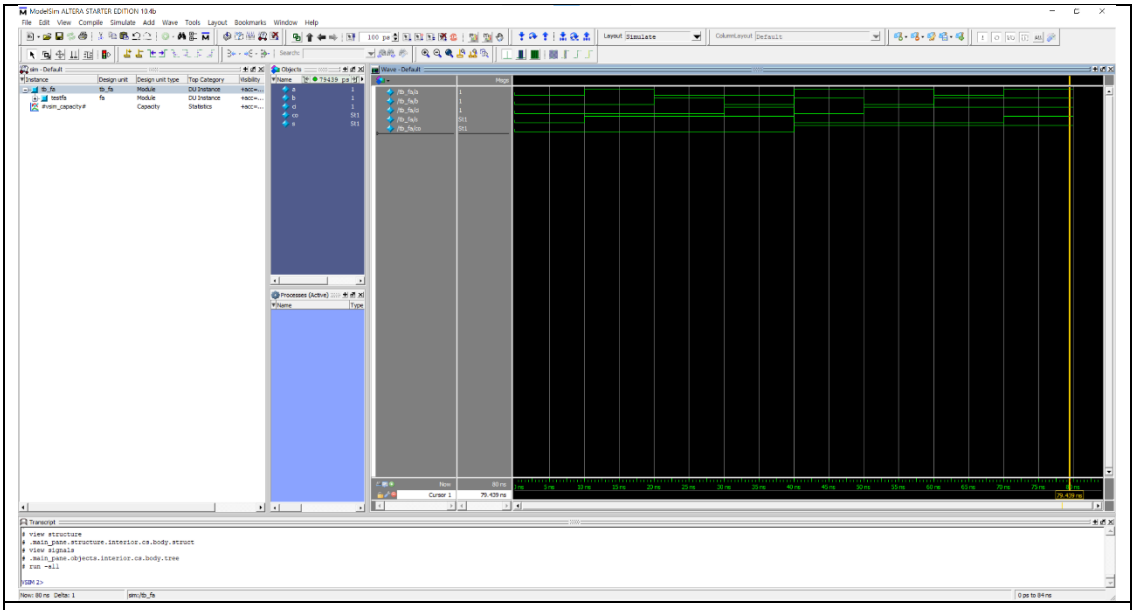
마지막 fa module에는 output s[3], co을 연결하고, input a[3], b[3]와 이전 fa module에서 넘어온 carry인 c[2]를 연결한다.

4. 설계 검증 및 실험 결과

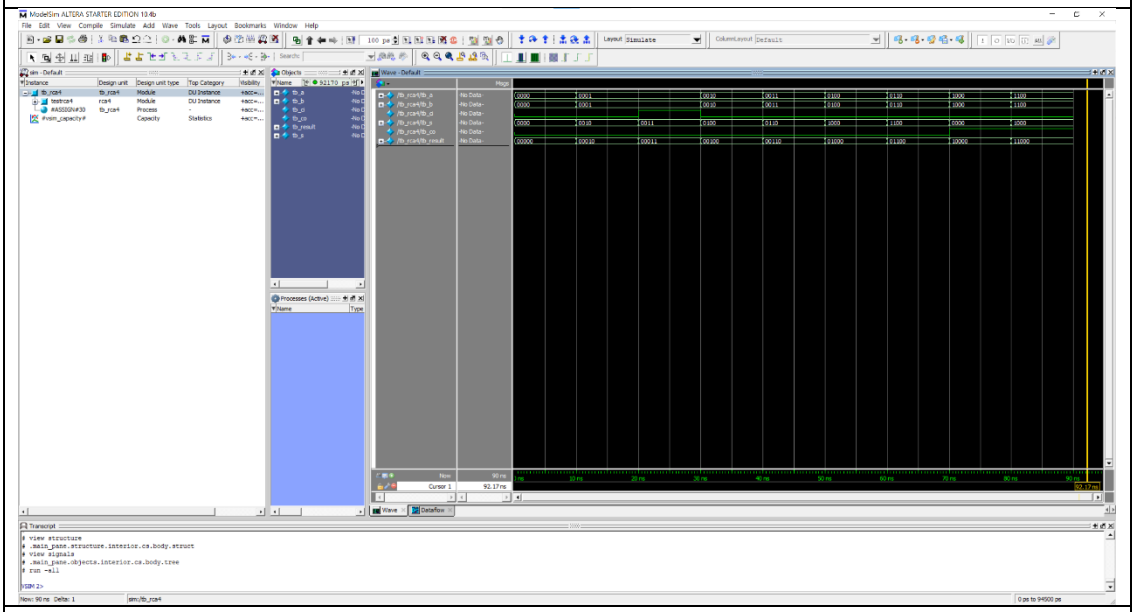
A. 시뮬레이션 결과



해당 testbench에서는 ha module의 a, b에 00, 01, 10, 11의 값을 #10 간격으로 넣었을 경우 나오는 결과다. s는 a와 b가 XOR의 경우에만 동작하고 있고, co은 AND의 경우에만 동작하고 있는 것을 확인할 수 있다.

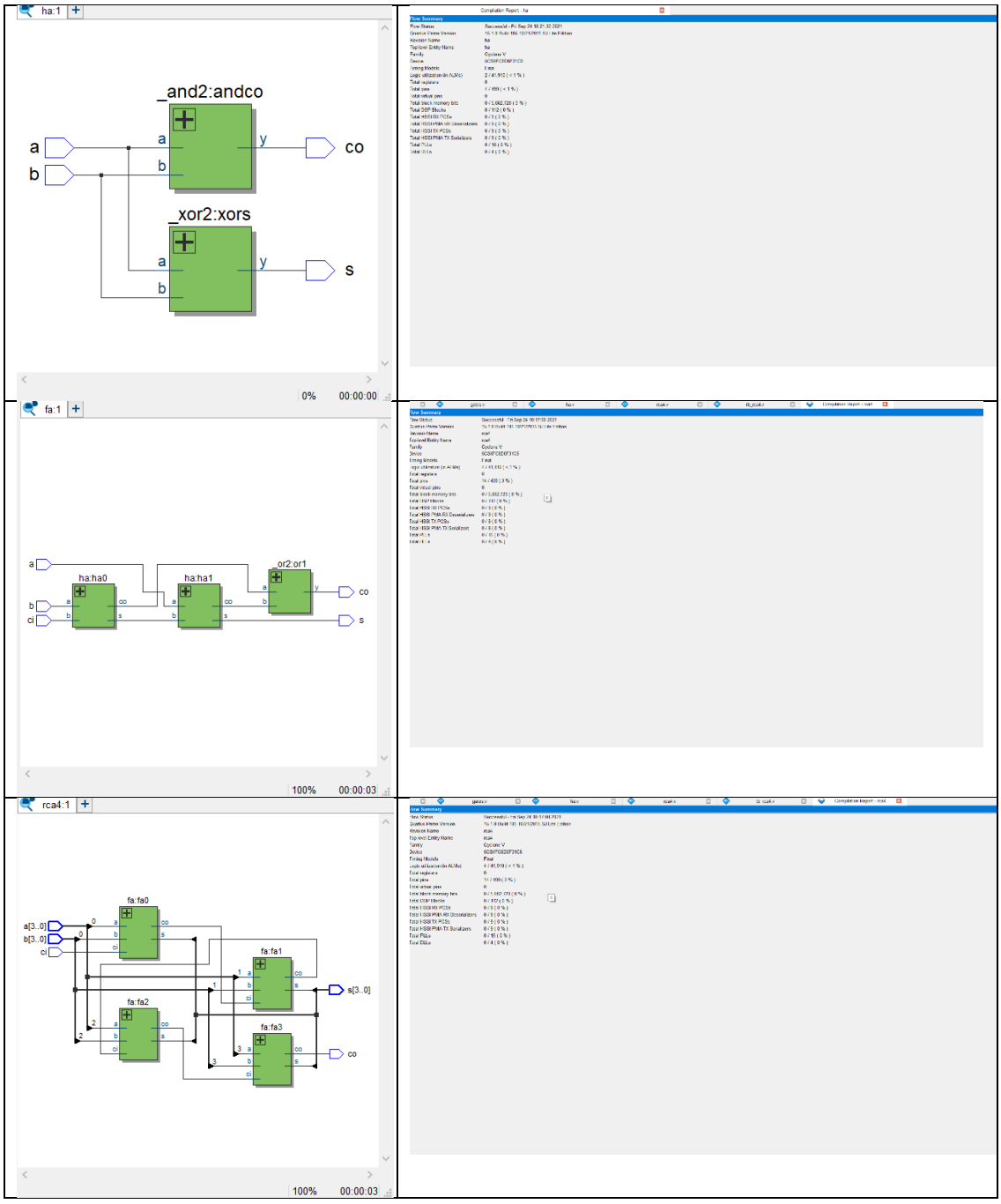


해당 testbench에서는 ha module의 a, b, c에 100, 010, 001, 110, 101, 011, 111의 값을 #10 간격으로 넣었을 경우 나오는 결과다. s는 a와 b, c가 XOR의 경우에만 동작하고 있고, co 또한 a, b, c 중 2개 이상이 켜졌을 경우 동작하는 모습을 확인할 수 있다.



해당 testbench는 Directed Verification을 통하여 rca4 module을 검증한다.
위의 fa의 testbench를 통해 fa는 정상적으로 동작함으로, 필자는 각 input과 output이 적절히 연결되었는지를 확인하여 해당 module이 정상적으로 설계되었는지 검증하였다.
a, b의 각 자리와 이전에서 발생한 carry를 각 자리마다 확인하여 정상적으로 다음 fa module에 넘어가는지 확인한다. 위의 결과를 통해 모든 값들의 input과 output이 정상적으로 연결되었음을 알 수 있다.

B. 합성(synthesis) 결과



해당 모듈의 설계를 확인하면 각각의 모듈이 정상적으로 설계되어 연결되었음을 알 수 있다.

C. FPGA board targeting 결과

5. 고찰 및 결론

A. 고찰

과제를 수행하면서 ha module과 fa module의 검증 없이 rca4 module을 testbench로 돌

려보았을 때 오류가 발생하였다. 이는 코드를 작성하면서 나온 오타에 의해 생긴 오류였다. 이러한 상황을 해결하기 위해서 ha module을 우선 testbench로 검증하고, fa module을 검증한 뒤 rca4의 testbench를 돌렸다.

B. 결론

해당 실험을 통하여 half adder와 full adder, 그리고 이를 통해서 만들 수 있는 RCA에 대해 자세히 알 수 있는 시간이었다. 또한 논리식을 정리해가면서 부울식 계산에 더욱 익숙해질 수 있었다. 그리고 검증을 할 때에는 low module을 먼저 검증한 후에 더욱 큰 level의 module을 검증해 나아가야하는 필요성을 느꼈다.

6. 참고문헌

이준환 교수님/디지털논리회로1/광운대학교(컴퓨터정보공학부)/2021
이준환 교수님/디지털논리회로2/광운대학교(컴퓨터정보공학부)/2021