컴퓨터구조실험 보고서

Verilog1 과제

과 목 컴퓨터구조실험

담당교수 이성원교수님

학 과 컴퓨터정보공학부

학 번 2021202058

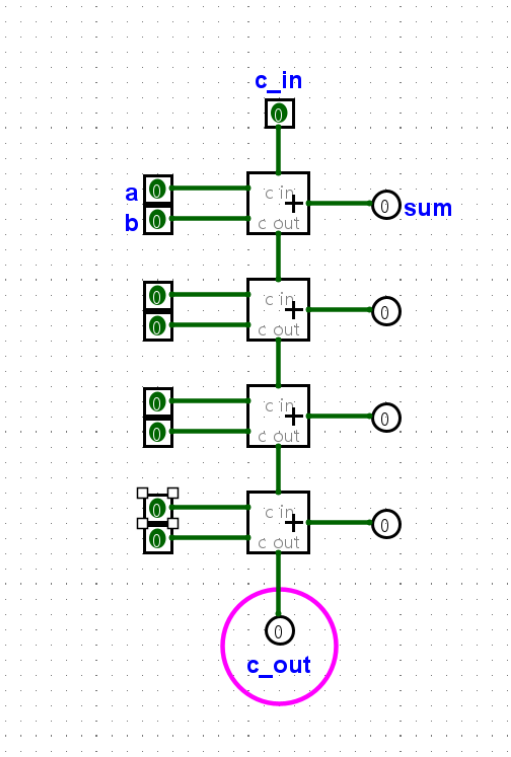
이 름 송채영

제 출 일 2021. 03. 19

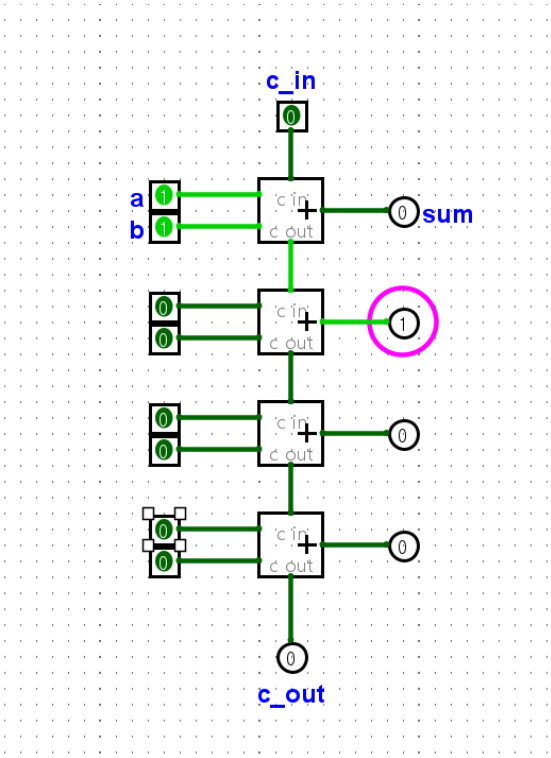
1. 결과 화면

1-1. Logism-evolution

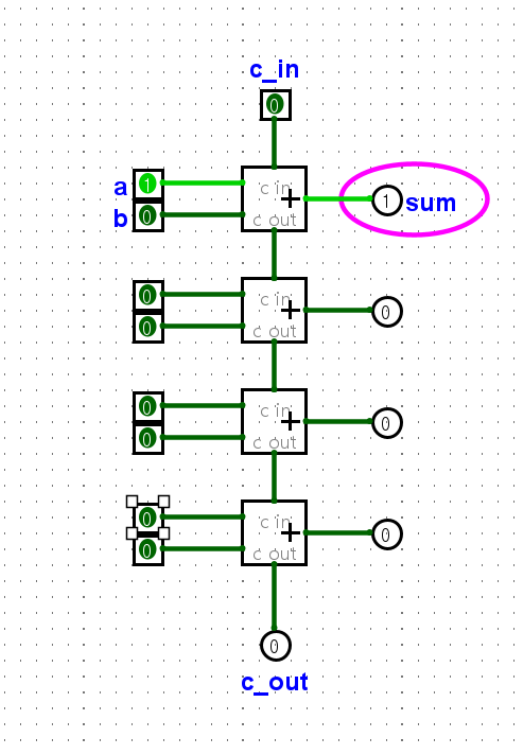
제공된 testbench 의 예시를 바탕으로 입력을 넣어보며 동작 결과를 확인하였다.



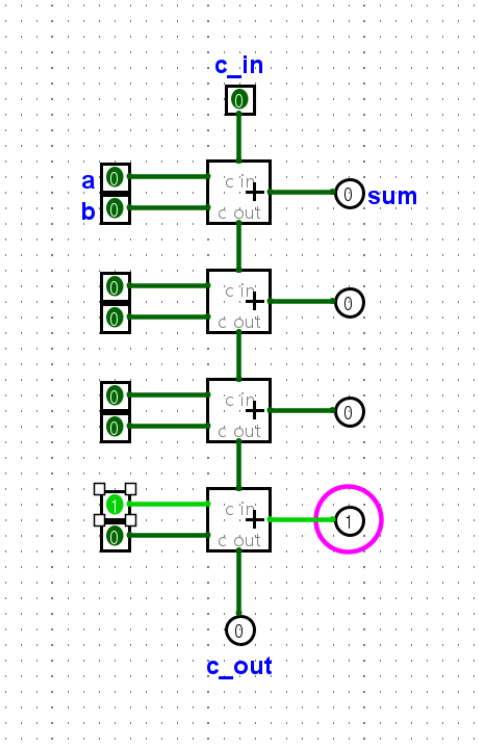
A 가 0 이고 b 가 0 일 때, 즉 0000, 0000 일 때 carry 가 발생하지 않으므로 c_out 은 0, sum 은 0 이 나오는 것을 알 수 있다.



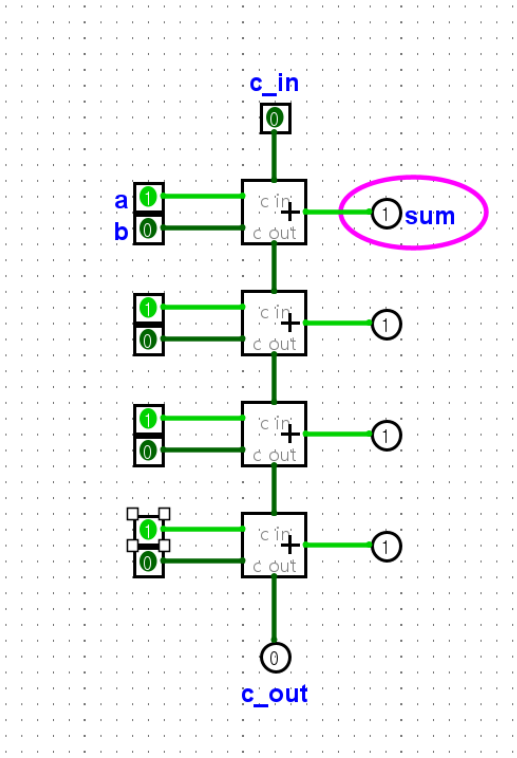
A 가 1 이고 b 가 1 일 때, 즉 0001, 0001 일 때 carry 가 발생하지 않으므로 c_out 은 0, sum 은 0002, 즉 2 가 나오는 것을 알 수 있다.



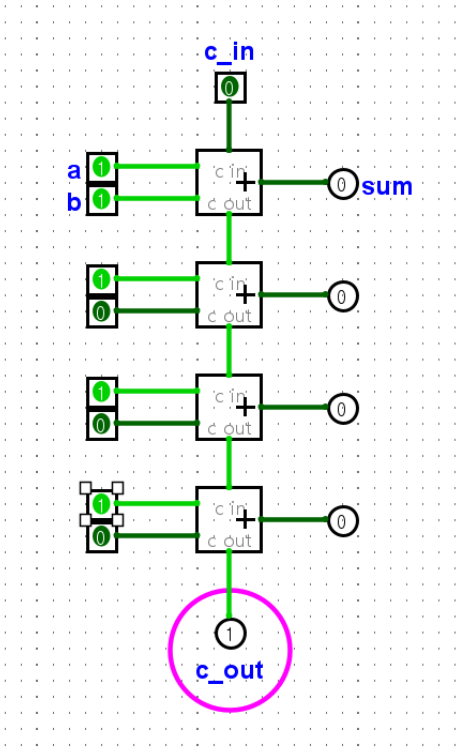
A 가 1 이고 b 가 0 일 때, 즉 0001, 0000 일 때 carry 가 발생하지 않으므로 c_out 은 0, sum 은 0001, 즉 1 이 나오는 것을 알 수 있다.



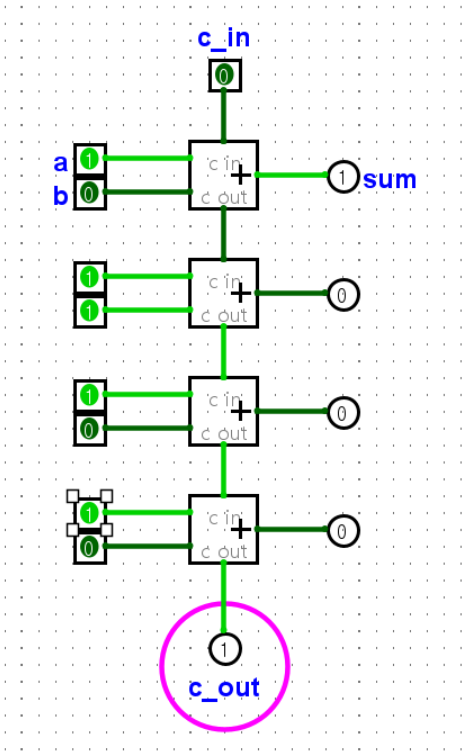
A 가 8 이고 b 가 0 일 때, 즉 1000, 0000 일 때 carry 가 발생하지 않으므로 c_out 은 0, sum 은 1000, 즉 8 이 나오는 것을 알 수 있다.



A 가 F 이고 b 가 0 일 때, 즉 1111, 0000 일 때 carry 가 발생하지 않으므로 c_out 은 0, sum 은 1111, 즉 F 가 나오는 것을 알 수 있다.



A 가 F 이고 b 가 1 일 때, 즉 1111, 0001 일 때 carry 가 발생하므로 c_out 은 1, sum 은 0000, 즉 0 이 나오는 것을 알 수 있다.



A 가 F 이고 b 가 2 일 때, 즉 1111, 0010 일 때 carry 가 발생하므로 c_out 은 1, sum 은 0001, 즉 1 이 나오는 것을 알 수 있다.

1-2. Lcarus Verilog

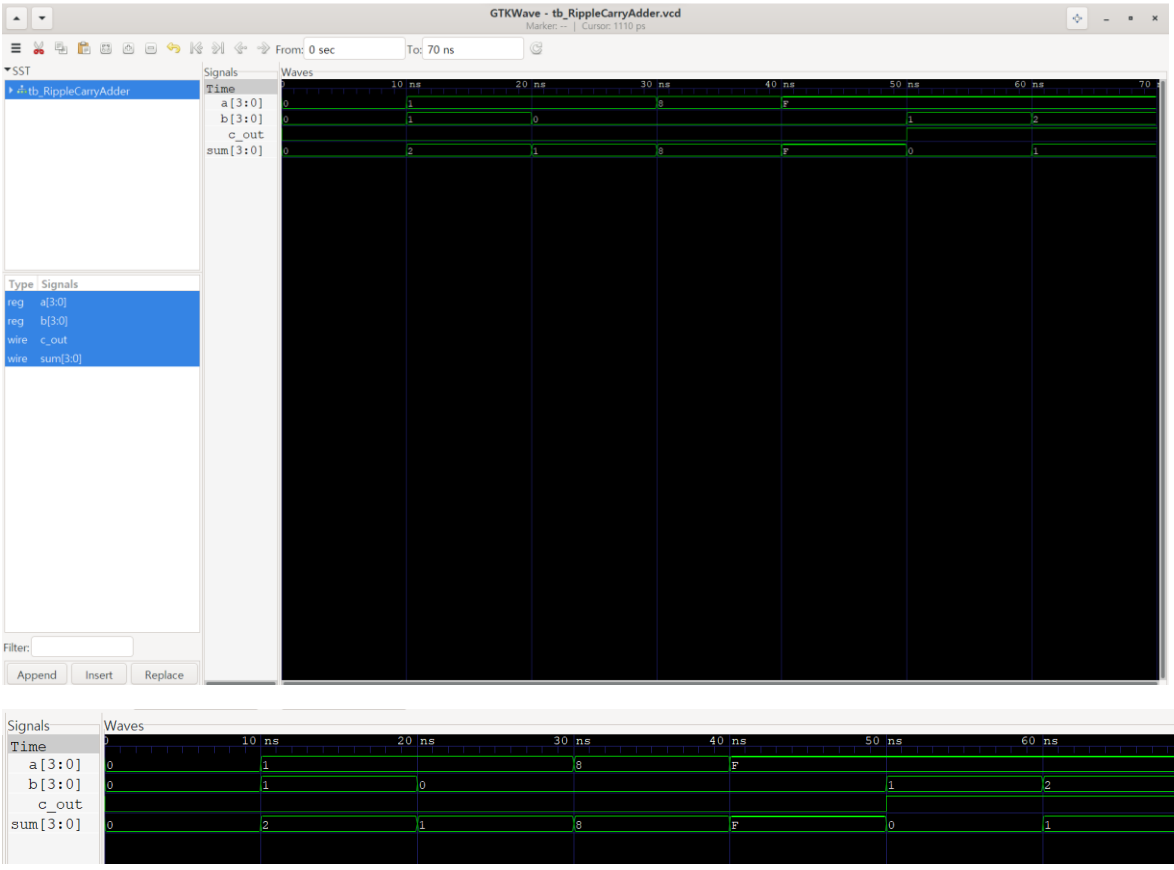
```
`include "FullAdder.v"

module RippleCarryAdder(a, b, c_out, sum);
      input [3:0] a, b;
      output c_out;
      output [3:0] sum;

      wire c0, c1, c2;

      FullAdder fa0(a[0], b[0], 1'b0, c0, sum[0]);
      FullAdder fa1(a[1], b[1], c0, c1, sum[1]);
      FullAdder fa2(a[2], b[2], c1, c2, sum[2]);
      FullAdder fa3(a[3], b[3], c2, c_out, sum[3]);
endmodule
```

위의 사진은 RippleCarryAdder 코드이며 아래 사진은 코드를 작성한 후 Icarus Verilog 로 컴파일 한 gtkwave 이다.



첫 번째 예시를 살펴보면, A 가 0 이고 b 가 0 일 때 carry 가 발생하지 않으므로 c_out 은 0, a 와 b 의 연산결과인 0000, 즉 0 이 sum 의 값으로 나온 것을 볼 수 있다. 두 번째 예시를 살펴보면 a 가 1 이고 b 가 1 일 때 carry 가 발생하지 않으므로 c_out 은 0, a 와

b 의 연산결과인 0010, 즉 2 가 sum 의 값으로 나온 것을 볼 수 있다. 세 번째 예시를 살펴보면 a 가 1 이고 b 가 0 일 때 carry 가 발생하지 않으므로 c_out 은 0, a 와 b 의 연산결과인 0001, 즉 1 이 sum 의 값으로 나온 것을 볼 수 있다. 네 번째 예시를 살펴보면 a 가 8 이고 b 가 0 일 때 carry 가 발생하지 않으므로 c_out 은 0, a 와 b 의 연산결과인 1000, 즉 8 이 sum 의 값으로 나온 것을 볼 수 있다. 다섯 번째 예시를 살펴보면 a 가 F 이고 b 가 0 일 때 carry 가 발생하지 않으므로 c_out 은 0, a 와 b 의 연산결과인 1111, 즉 F 가 sum 의 값으로 나온 것을 볼 수 있다. 여섯 번째 예시를 살펴보면 a 가 F 이고 b 가 1 일 때 carry 가 발생했으므로 c_out 은 1, a 와 b 의 연산결과인 0000, 즉 0 이 sum 의 값으로 나온 것을 볼 수 있다. 마지막 예시를 살펴보면 a 가 F 이고 b 가 2 일 때 carry 가 발생했으므로 c_out 은 1, a 와 b 의 연산결과인 0001, 즉 1 이 sum 의 값으로 나온 것을 볼 수 있다. 1-1 과제인 Logisim-evolution 의 결과와 같게 나오므로 코드를 잘 구현하였음을 알 수 있다.

2. 고찰

1-bit full adder 의 코드를 구현한 후 RippleCarryAdder 의 코드에서 instance 하기 위해 같은 directory 에 위치하게끔 하면 되는 줄 알았다.

```
C:\Users\a0109\OneDrive\바탕 화면\3-1\컴구실\verilog_code>iveril
og -Wimplicit -o tb_RilppleCarryAdder.o tb_RippleCarryAdder.v
./RippleCarryAdder.v:8: warning: macro b0 undefined (and assumed
 null) at this point.
./RippleCarryAdder.v:8: error: Unknown module type: FullAdder
./RippleCarryAdder.v:9: error: Unknown module type: FullAdder
./RippleCarryAdder.v:10: error: Unknown module type: FullAdder
./RippleCarryAdder.v:11: error: Unknown module type: FullAdder
5 error(s) during elaboration.
*** These modules were missing:
        FullAdder referenced 4 times.
***
```

하지만 다음과 같은 error 가 떠 제공해주신 testbench 코드를 확인해보니 같은 directory 에 위치해야 하는 것 뿐만 아니라 `include "FullAdder.v" 이런 식으로 import 해주어야 한다는 것을 알았다. cmd 에서 코드를 돌려보는 과정에서 구문 오류가 났다는 error 도 떠 코드를 다시 확인해보니 `, ' 이 두개를 반대로 써서 생긴 오류였다. 2 학년 2 학기, 컴퓨터기초실험 2 과목에서 다 배우고 실습했던 것들인데, 다시 해보니 잘 기억이 안 나 복습이 필요하다는 것을 느꼈다.

3. Reference

컴퓨터구조실험 강의자료