Hardware System Design, Lab 03

2013-11392 김지현

# 구현체 설명

## my\_add 구현 설명

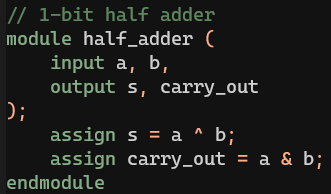


Figure 1. 1-bit half adder

1-bit half adder를 가장 먼저 구현했다. 1-bit half adder는 두개의 1bit 정수의 합인 a+b를 계산해, 덧셈 결과와 carry를 출력하는 모듈이다. XOR Gate 하나와 AND Gate 하나를 사용해 1-bit half adder를 쉽게 구현할 수 있었다.

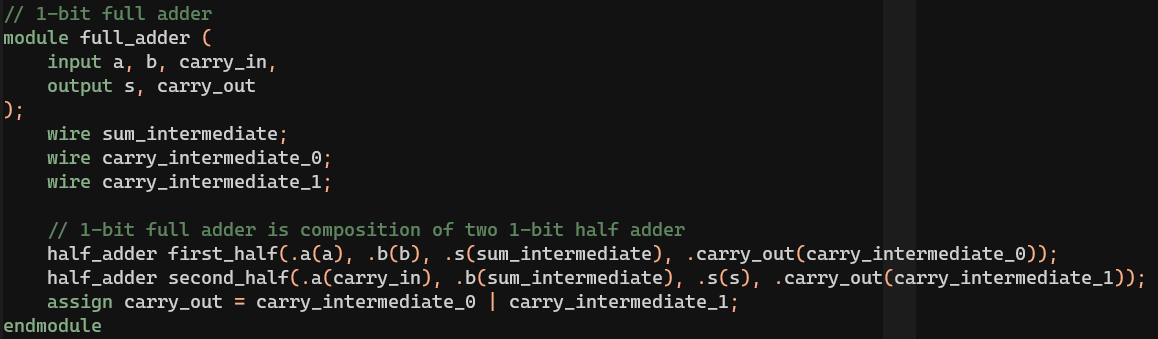


Figure 2. 1-bit full adder

그 다음으로는 1-bit full adder를 구현하였다. 1-bit full adder는 carry\_in 을 입력으로 받을 수 있는 adder로, “a+b+carry\_in”을 계산하여 덧셈 결과와 carry를 출력하는 모듈이다. 1-bit half adder 두개를 사용해 “a+b”를 먼저 계산하고, “(a+b)+carry\_in”을 계산하는 방식으로 구현하였다.

N-bit full adder는 위의 half\_adder 하나와 full\_adder 여럿을 조합하여 구현하였다. 사람이 손으로 여러 자리 덧셈을 하는것과 동일한 방식으로, 가장 낮은 자리수부터 더하기 시작해 발생하는 carry를 위 자리수로 올리는 방식으로 구현하였다. 덧셈을 하다보면 가장 마지막 자리수의 carry가 발생하게 되는데 이는 overflow로 출력하였다.

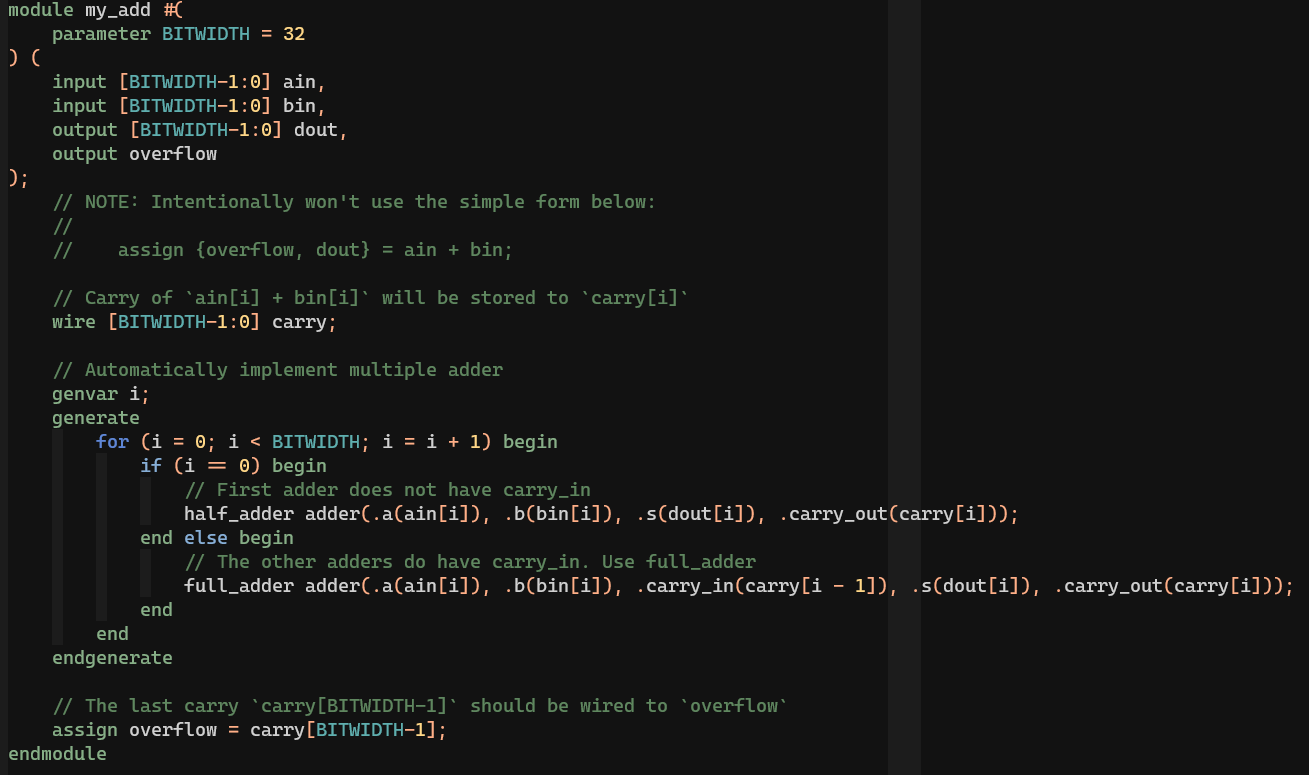


Figure 3. N-bit full adder

## my\_mul 구현 설명

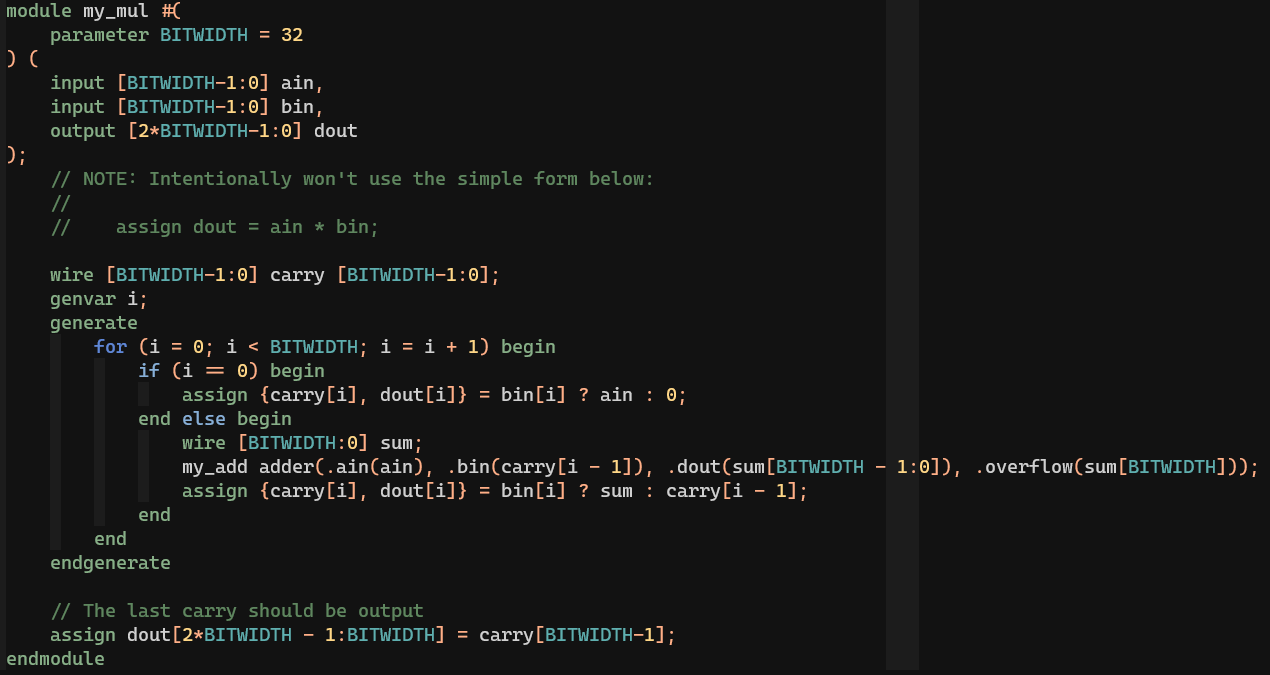


Figure 4. N-bit unsigned int multiplier

길이가 BITWIDTH인 벡터 “bin”의 0번째 항부터 BITWIDTH-1번째 항까지 차례대로 순회하며, “bin”의 각 비트에 ain을 곱하는 방식으로 구현하였다. 이 때 이전에 곱했던 결과를 다음 bit 곱셈할때에 더해줘야 하는데, 이 과정에서 my\_add를 사용하였다.

## my\_fusedmult 구현 설명

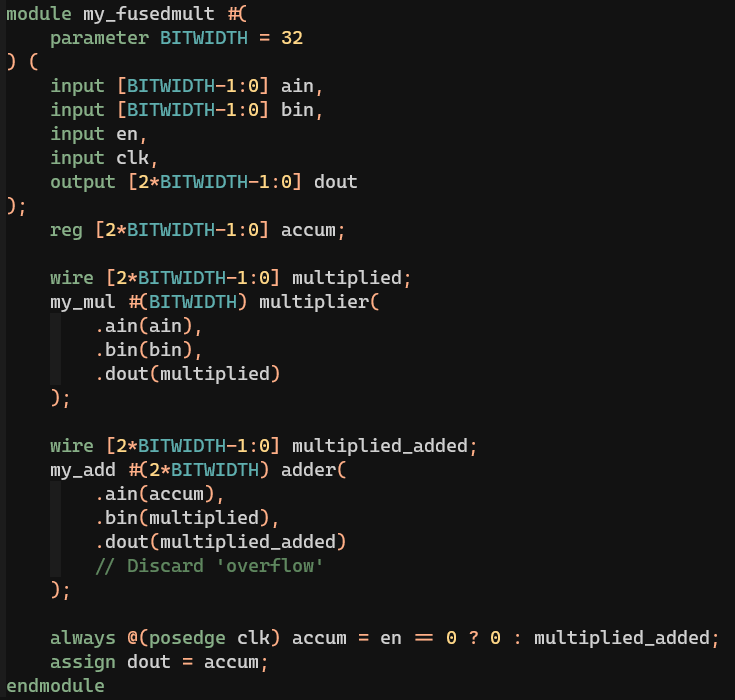


Figure 5. my\_fusedmult

My\_fusedmult는 매 Clock tick 마다 en이 0일 경우 0을 출력하고, en이 1일 경우 이전의 결과에 ain\*bin을 더하여 출력하는 함수이다. Verilog의 레지스터 기능과 “always @(posedge clk)” 문법을 사용하여 어렵지 않게 구현할 수 있다. 이 때 덧셈과 곱셈을 하기 위해 Verilog에 내장된 덧셈/곱셈 문법이 아닌 my\_add와 my\_mul 모듈을 사용하였다.

과제 Spec에 Rising Clock Edge를 사용해야하는지, Falling Clock Edge를 사용해야하는지 명시되어있지 않았는데, Rising Clock Edge를 사용하도록 구현했다. 또한 Rising Clock Edge 일 때에만 ain, bin, en과 같은 입력을 읽도록 구현하였다.

# 실험 결과 분석

Figure 6. Using "is\_ok" for test

과제 요구조건으로 “주어진 testbench를 사용할 것” 이 있어서 제출된 코드에는 포함되어있지 않지만, 올바르게 동작하는지 테스트하기 위해 사진과 같이 my\_add, my\_mul, my\_fusedmult와 Verilog의 내장 덧셈, 곱셈 기능의 결과를 비교하였다. Verilog 내장 덧셈, 곱셈 기능은 올바르게 동작하는 것이 보장되므로 편리하게 실행 결과를 확인할 수 있었다.

그 외에도 아래와 같이 주어진 testbench로 my\_add, my\_mul, my\_fusedmult의 동작을 테스트하였고, 올바르게 동작하는 것을 확인할 수 있었다.

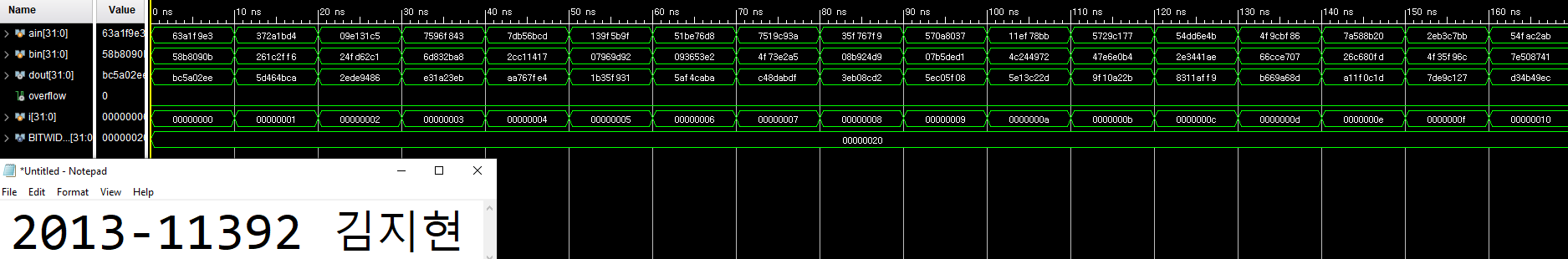


Figure 7. Testing my\_add

my\_add와 같은 경우 주어진 testbench를 그대로 실행하면 overflow의 동작여부를 확인할 수 없어, 고친 testbench로 overflow의 정상동작 여부를 따로 체크해줬다.

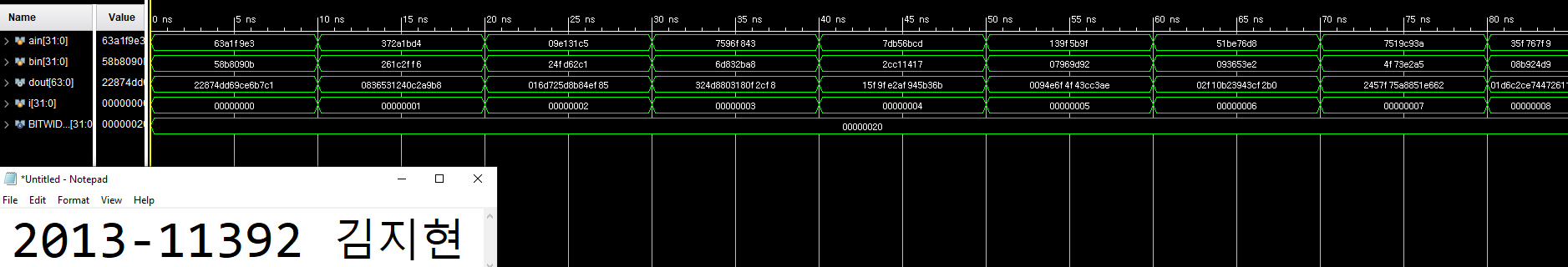


Figure 8. Testing my\_mul

my\_mul은 정상적으로 동작했다는 점 이외에 특기할만한 부분은 없었다.

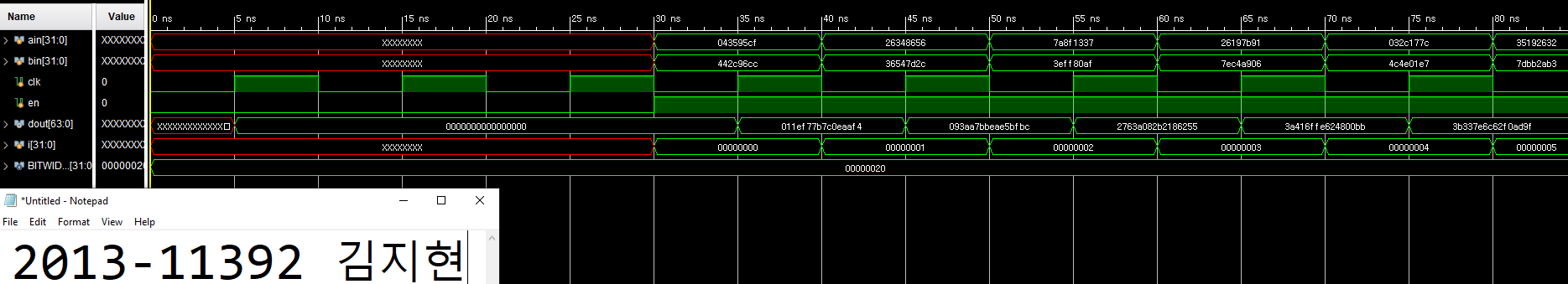


Figure 9. Testing my\_fusedmult

my\_fusedmult는 Rising Clock Edge마다 출력이 정상적으로 업데이트되는 것을 확인할 수 있었다. 테스트 맨 처음에 dout이 에러를 출력하는 것을 볼 수 있는데, Rising Clock Edge를 한번도 만난적이 없으면 내부 상태가 초기화되지 않게 작성했으므로 의도한 결과이다.

# 결론

Verilog 코딩을 통해 adder와 같은 작은 회로를 구현하는법을 이번 Lab을 통해 익힐 수 있었고, 코드가 올바르게 동작하는지 테스트하는법 또한 익힐 수 있었다. 그리고 단순한 Combinational Logic 뿐만이 아닌 Clock tick에 맞추어 실행되는 Sequential Logic을 어떻게 Verilog로 작성하는지 또한 배울 수 있었다.