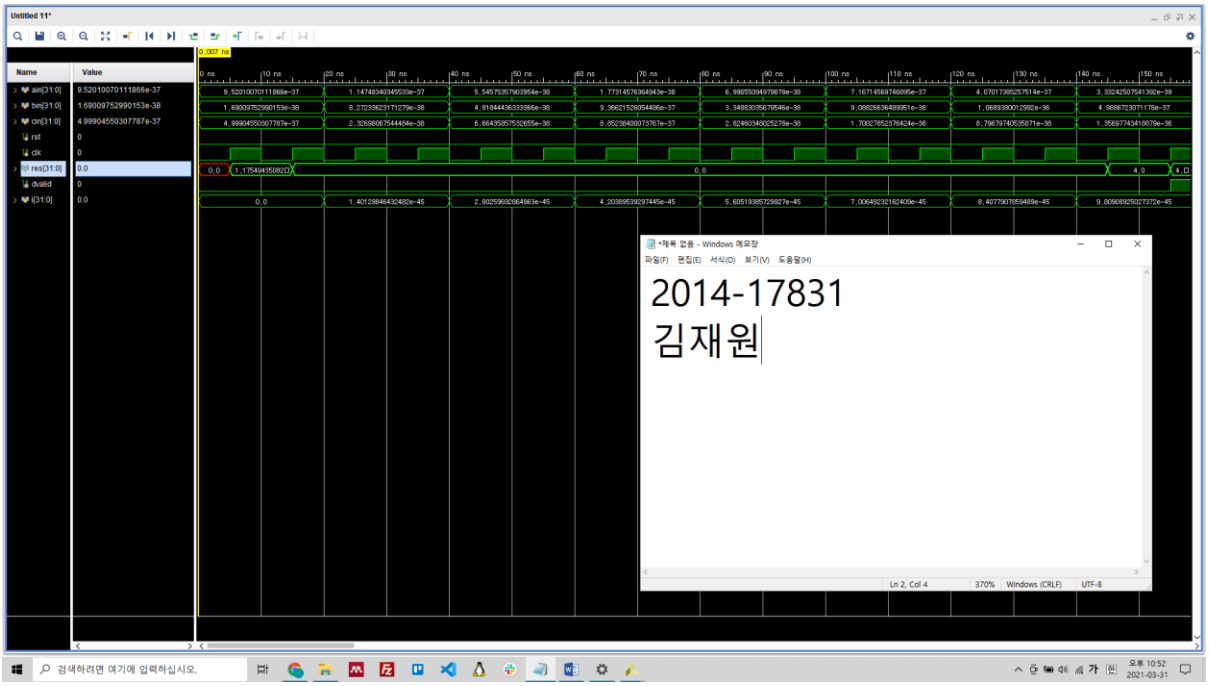
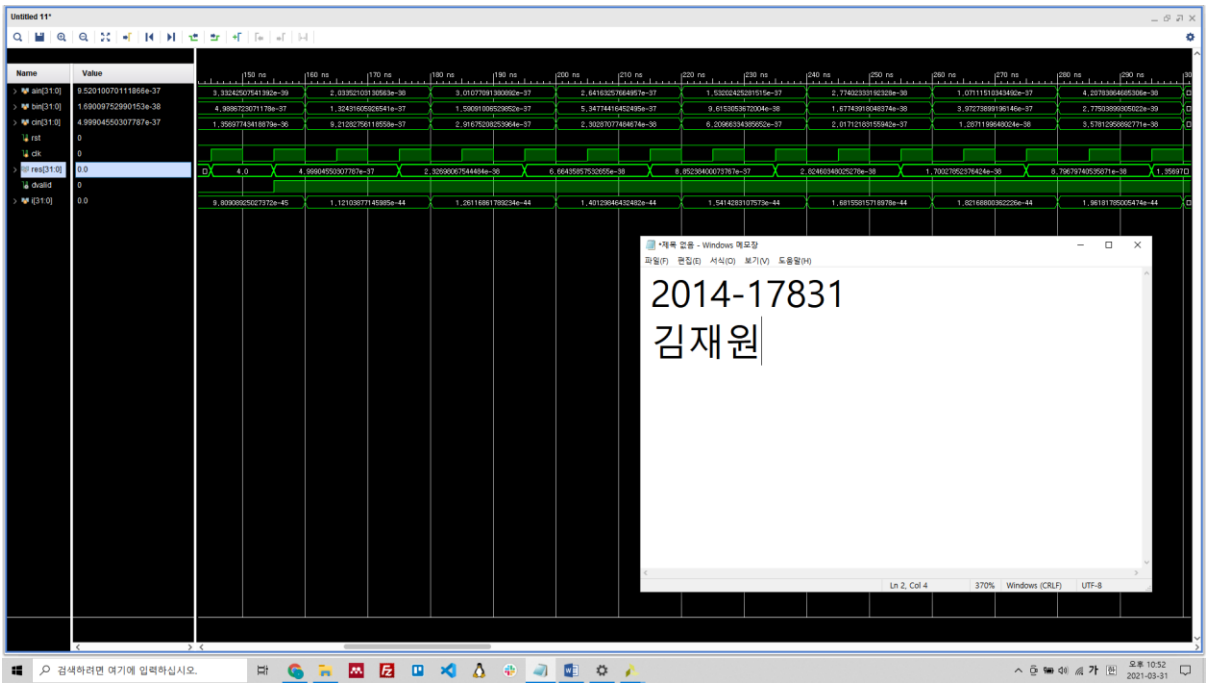
2014-17831 김재원

<Floating Point Multiply-Adder>

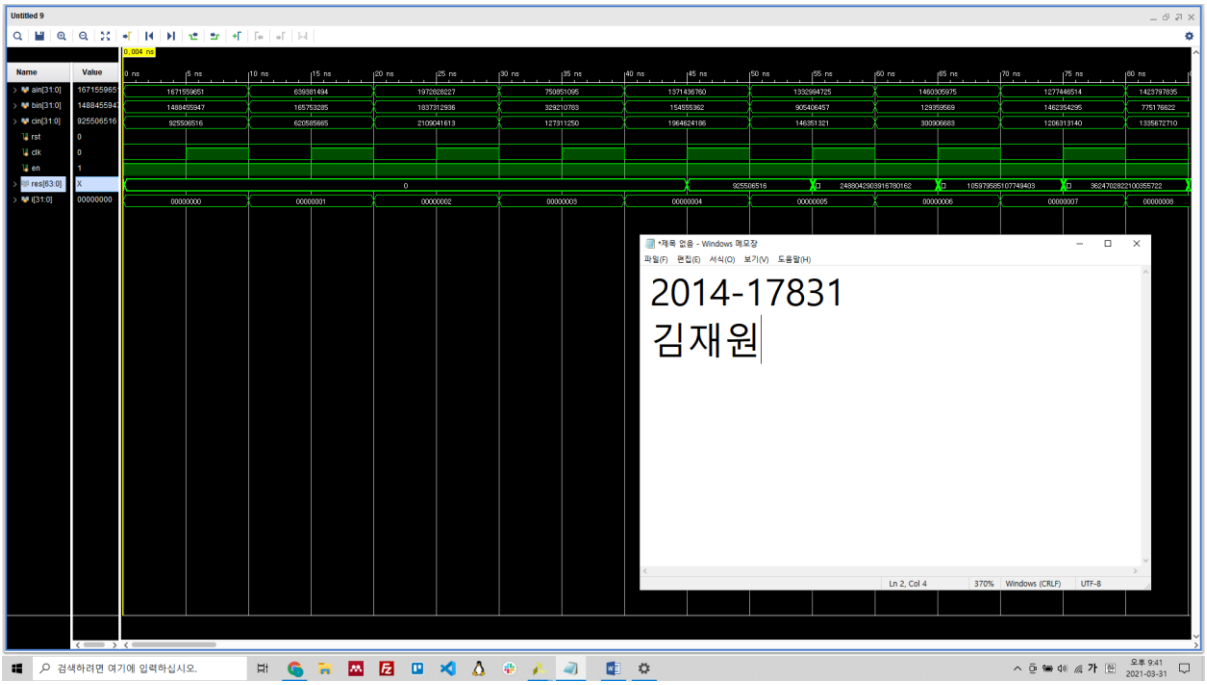




- Floating point multiply_adder에 대한 test bench를 수정한 코드에 대한 waveform simulation 결과입니다.
- ain, bin, 그리고 cin 값에 대해 주어진 웹사이트를 참고하여 적절한 값을 넣고자 했습니다. 의도했던 것은 denormalized range에 속하지 않으면서 2^4보다 작은 값의 floating point number을 임의로 만들어 결과를 확인하고자 하였습니다. 그러나 아직까지 제대로 파악하

지 못한 이유로 ain = {6'b10000, $urandom%(2**5), 20'b000...}와 같은 방식으로 구현하였을 때 앞부분에 원치 않는 0들이 추가되어 매우 작은 denormalized 범위의 소수가 만들어졌습니다. 해당 문제를 해결하지 못한 채 우선 multiply_adder가 잘 작동하는지 확인하였고, 결과값이 일정한 delay 후 잘 출력되는 것을 확인하였습니다.

<Integer Multiply-Adder>

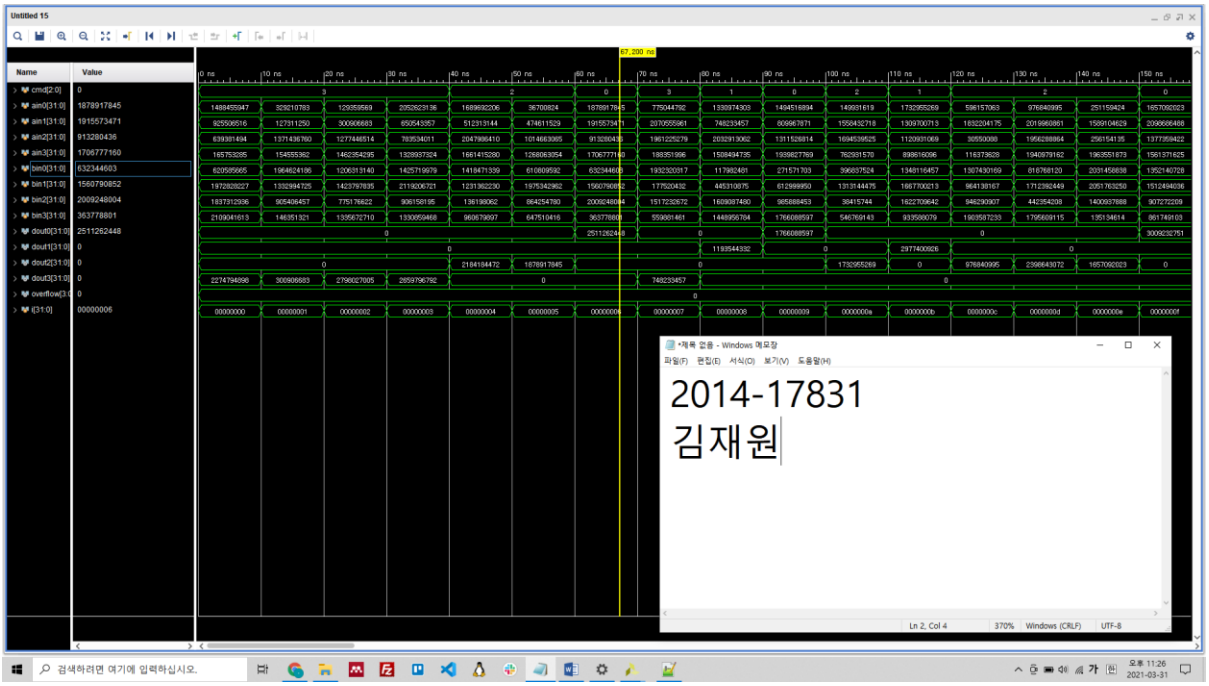


- 32bit multiply-adder을 구현한 코드에 대한 waveform simulation입니다.
- IP catalog에서 주어지는 BaseIP의 multiply_adder을 이용하여 구현하였고, A, B, C는 모두 32bit unsigned integer로 지정하였습니다. Output에 해당하는 P는 연산 결과에 overflow가 있을 경우 multiply_adder이 제대로 동작하는지 확인할 수 없을 것을 우려하여 64bit으로 지정하였습니다. (이 부분에 대해서는 구현 시 자율적으로 선택할 수 있는 것으로 질의응답을 통해 이해했습니다.)
- Test bench를 작성하기 위해 input signal과 parameter에 대해 파악하고자 documentation을 참고하였습니다. Clock enable signal, clock, reset, 등 input이 3주차에 구현했던 fused multiplier나 IP catalog를 이용해 구현한 floating point multiply-adder의 test bench와 유사한 부분이 많아 해당 코드와 거의 동일하게 작성하였습니다. 다만, floating point IP catalog에 대한 test bench에서는 reset signal이 .aresetn(~rst)으로 되어있었는데, 해당 부분에 대해 동일하게 구현하니 output P가 전부 0으로 출력되었습니다. 그래서 floating point operation에 대한 IP catalog documentation을 보니 aresetn이 예외적으로 active low signal이라는 것을 확인할 수 있었고, 이에 integer multiply_adder의 reset signal

은 .SCLR(rst)이라고 명시해주었습니다.

- 출력 결과의 경우 (A * B) + C를 연산한 결과, 즉 (ain * bin) + cin의 값이 P (res)에 잘 출력되는 것을 확인할 수 있었습니다.

<Adder Array>



- genvar을 이용해 만든 adder_array에 대한 test bench 코드를 실행한 결과입니다.
- 구현 방식은 다음과 같습니다:
  - 우선, adder array 안의 모든 input과 output을 관리해줄 wire vector을 선언하였습니다. genvar로 만들어지는 여러 개의 adder 각각에 대한 input과 output을 연결해야 했기에 32bit wire 4개로 이루어진 vector 형태로 선언하였습니다. 이 wire들은 실제 adder_array의 input/output에 해당하는 ainX, binX, doutX를 adder와 연결해주어 input을 받아왔습니다.
  - 추가로, assign {dout[0], dout[1], dout[2], dout[3]} = {dout0, dout1, dout2, dout3}; 의 경우 과제 안내에 따르면 dout[i] 값을 adder로부터 받아와서 사용하는 것이 아니라 ainX+binX 값을 직접 계산해서 사용하도록 되어있기에 꼭 필요하지 않은 것 같았지만, addition operation이 중복되어 일어난다는 점에서 dout[i]과 doutX를 연결해두고 dout[i] 값을 사용해 doutX을 값을 assign하는 것이 더 나을 것 같다는 생각이 들었습니다.

- Test bench 구현 방식의 경우 my_add의 test bench와 거의 동일하여 따로 설명할 것이 없는 것 같습니다. Test bench의 동작을 보면, cmd 값에 따라 doutX이 잘 바뀌는 것을 확인할 수 있습니다. 예를 들어, 노란색 커서가 놓여있는 곳을 보면 cmd 값이 0인데, dout0의 값만 ain0과 bin0을 합한 값으로 바뀌어 있고, 나머지는 모두 0으로 출력되어 있습니다. Overflow는 my_add의 경우와 마찬가지로 전부 0으로 출력되었습니다.