**Hardware System Design**

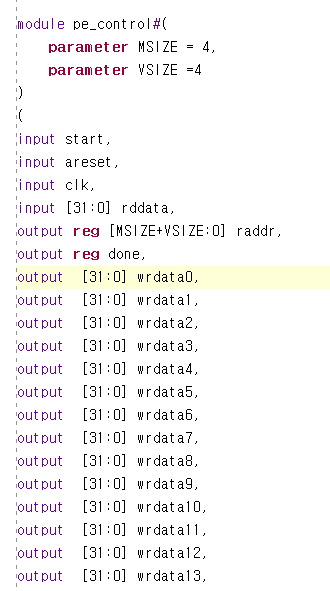
**Term project V0 submission**

**이다운 2016-13919**

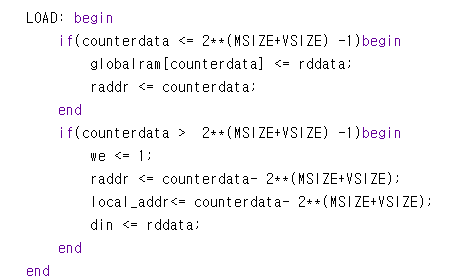
0. Overview

lab06 에서 V x V 연산을 위한 pe\_controller를 구현하였다. 이를 수정 및 보안해서 M x V 연산이 가능하게 만드는 것이 이번 Term project V0 submission의 목표이다.

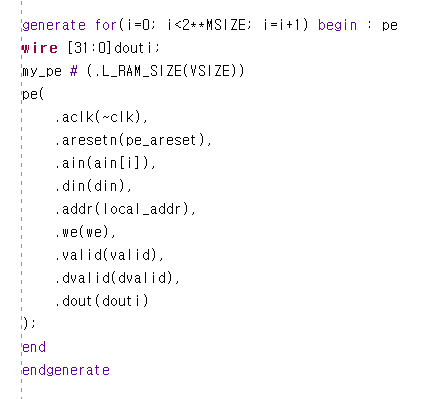
1. Code



위 사진은 모듈의 parameter와 port 코드 사진이다. parameter 에서 MSIZE, VSIZE는 각각 Matrix의 크기를 의미하는데 Matrix의 row가 2MSIZE, column이 2VSIZE 라는 의미이다. PORT는 lab06 과 비교하면 output만 다른데 M x V의 output으로 1 x 2VSIZE Matrix를 출력해야 한다. 그래서 1 x 2VSIZE Matrix 의 원소 값을 동시에 출력하게 여러 output port를 두었다. 이 report에서는 MSIZE, VSIZE 둘 다 4로 가정하고 할 것 이므로 16개의 wrdata output을 만들었다.

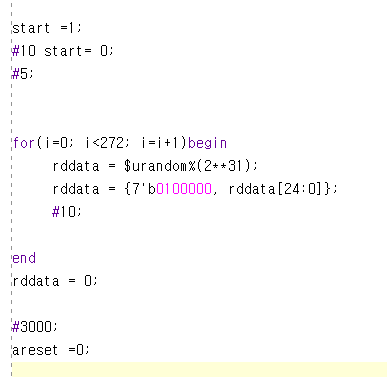


기본적인 뼈대는 lab06과 같다. LOAD단계에서 M의 원소 2MSIZE+VSIZE 개 받고 V의 원소 2VSIZE개를 받아 MY\_PE의 local 버퍼에 저장해준다.



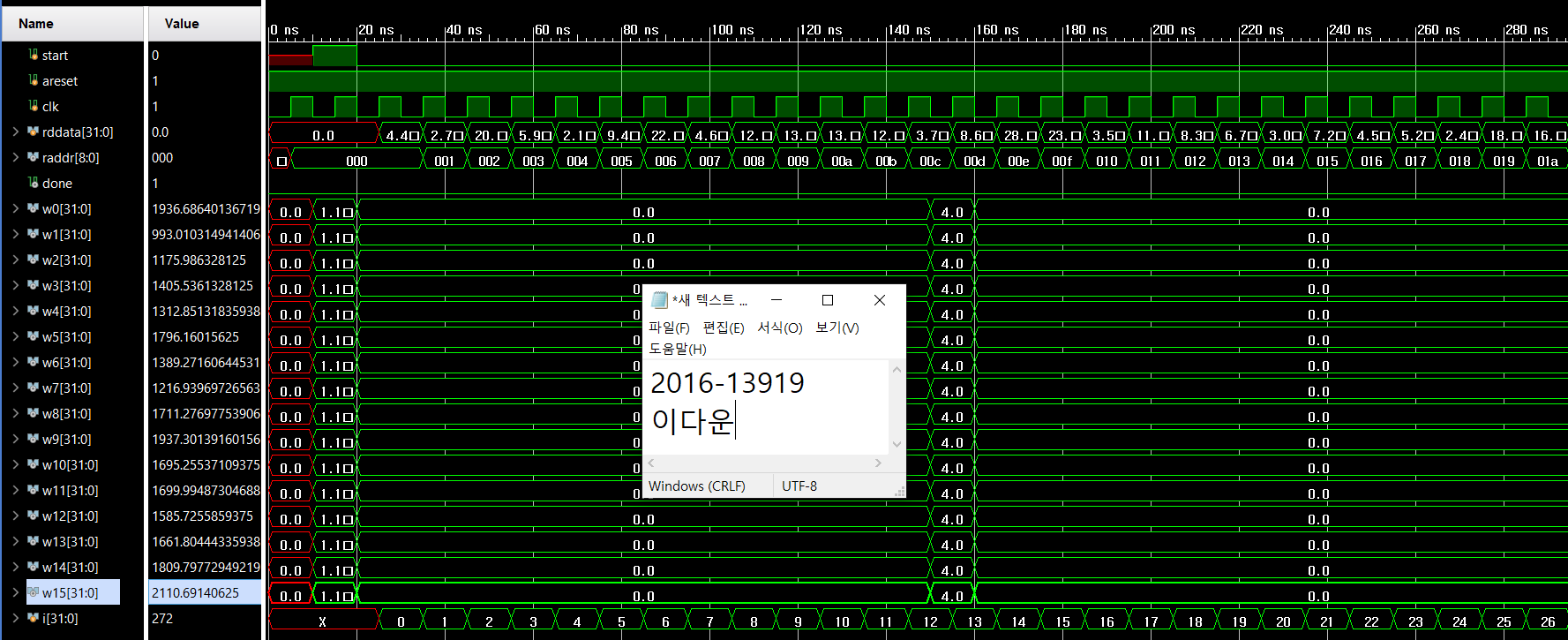
그리고 행렬 곱셈의 각 행들의 연산을 동시에 할 수 있도록 generate for문으로 2MSIZE개의 MY\_PE모듈을 만들어줬다. 각 MY\_PE[I]의 output port인 douti는 전체 모듈 output port와 연결될 것이다.

3. Testbench & Simulation



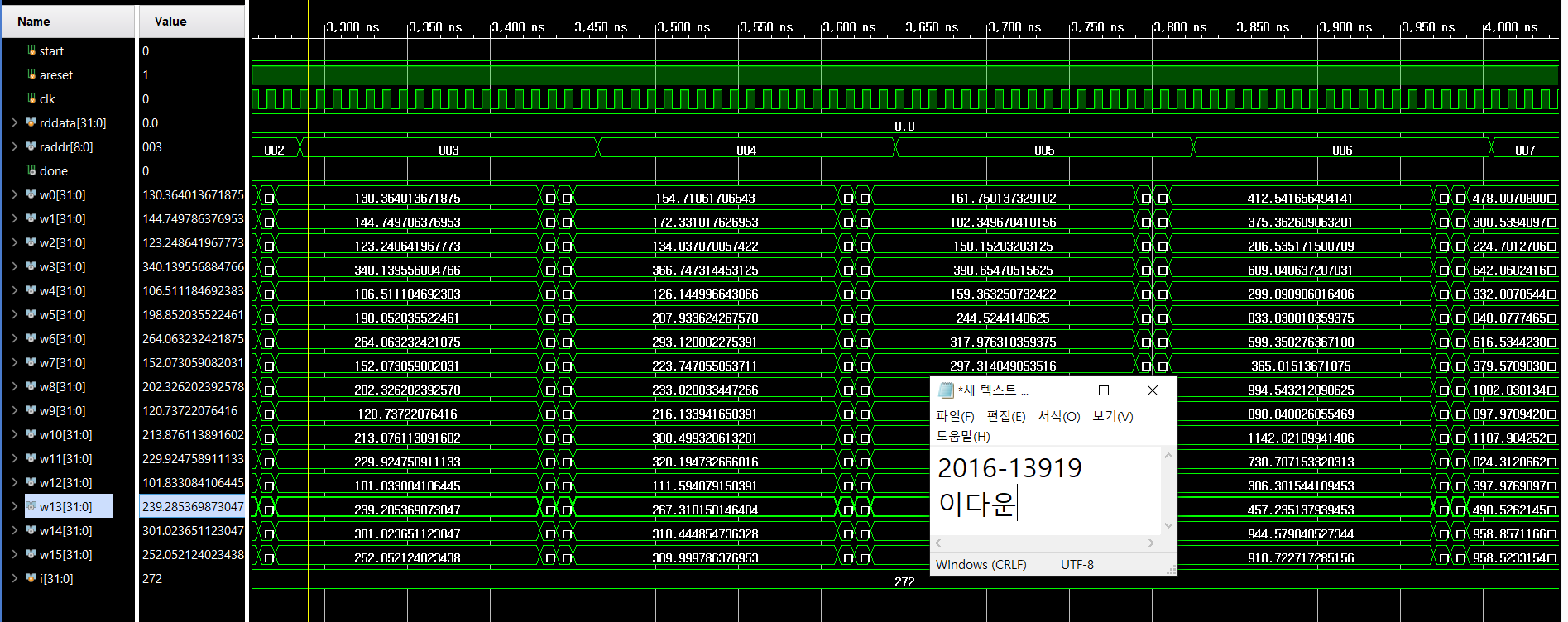
Simulation은 MV연산 크기를 (16x16) X 16 연산으로 진행할 것이다.

testbench는 start신호를 준 다음에 input으로 M,V의 원소 총 272개를 매 클럭마다 차례대로 입력해주었다. 이때 $urandom 함수에서 너무 큰 수가 나오면 overflow가 나올 수 있으니 부동소수점의 exp부분을 조정해 주었다.

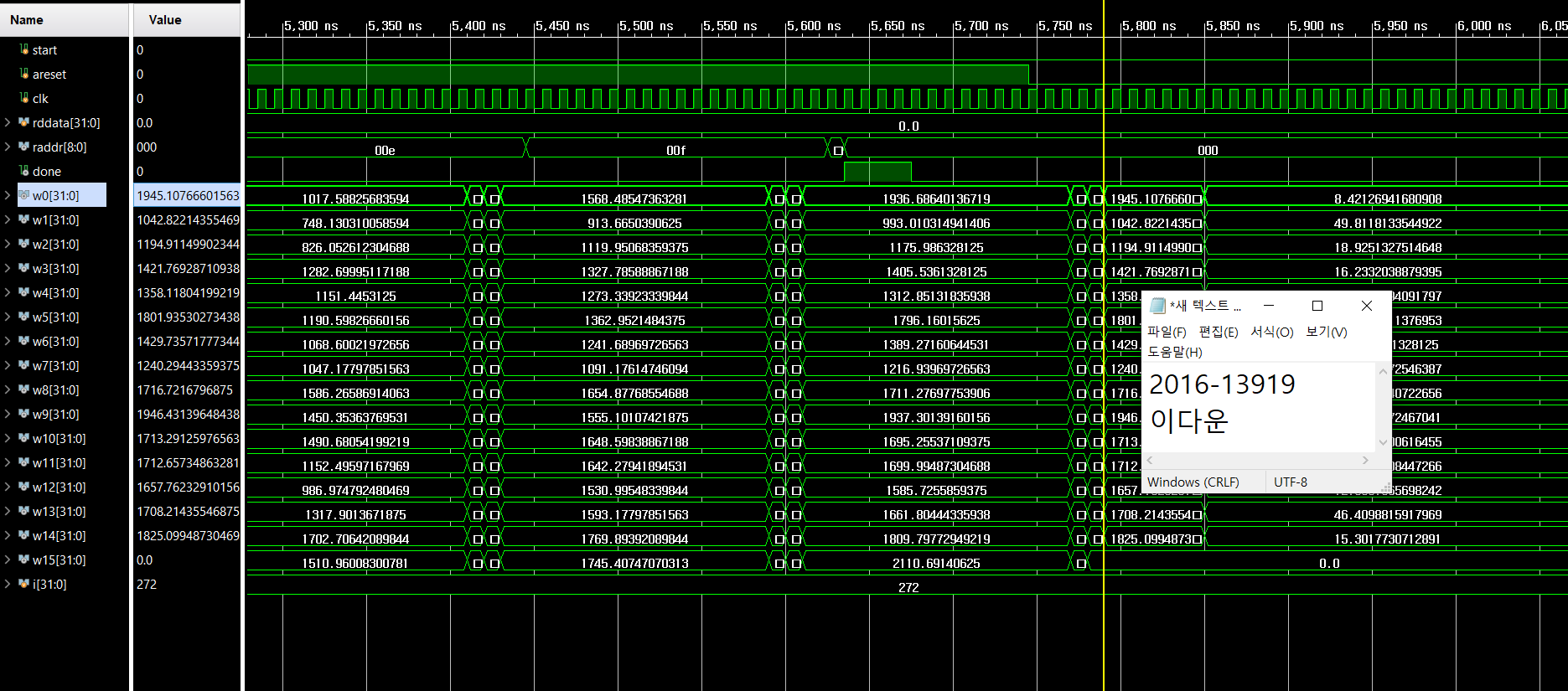


(확대하면 보입니다.)

simulation의 초기 부분이다. start신호가 1이 되고 rddata 포트로 Matrix의 원소가 될 값들이 입력되고 있는 것을 확인할 수 있다.



CALC state 중간의 스크린샷으로 raddr의 값에 해당하는 위치까지의 결과를 확인할 수 있다.



마지막으로 CALC state가 끝나고 DONE state로 들어가는 것을 확인할 수 있다. 모든 연산이 끝난 뒤 DONE 시그널이 1이 된다. done이 1일때의 output값이 MxV의 연산 결과 값이다.

4. Discussion

MxV 원소를 읽어올 때 readmemh 함수를 사용하면 훨씬 빠르게 LOAD단계가 끝날 것이다. 이를 사용하여 수행시간을 줄일 수 있을 것 이다.

Submission에서는 MxV의 연산만 보이면 돼서 최적화에 대해서는 고려하지 않았다.

output을 각 포트마다 출력해야 하는지 writememh함수를 사용 해야 하는지 아직 확실하게 모르겠어서 일단 port로 출력하는 방식을 택했다.