7주차 결과보고서

전공: 컴퓨터공학과 학년: 3학년 학번: 20191612 이름: 윤기웅

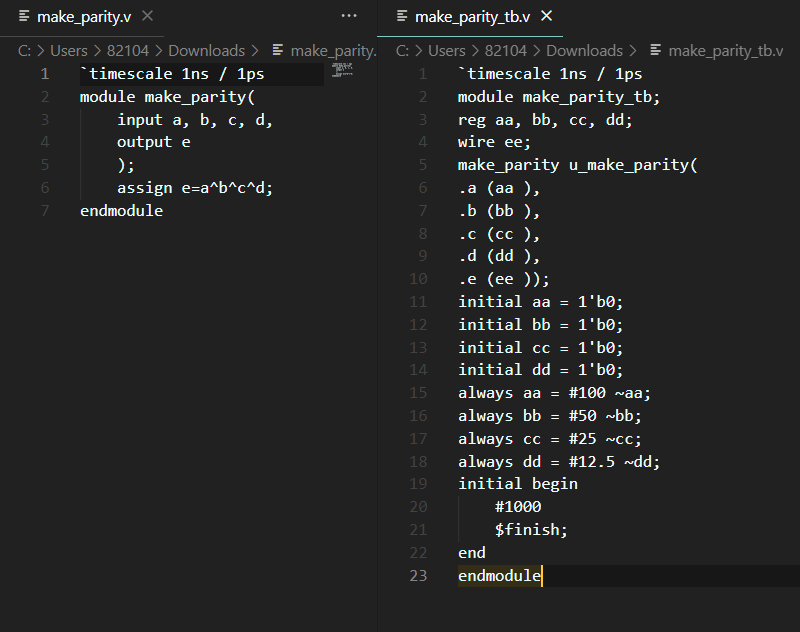
1.

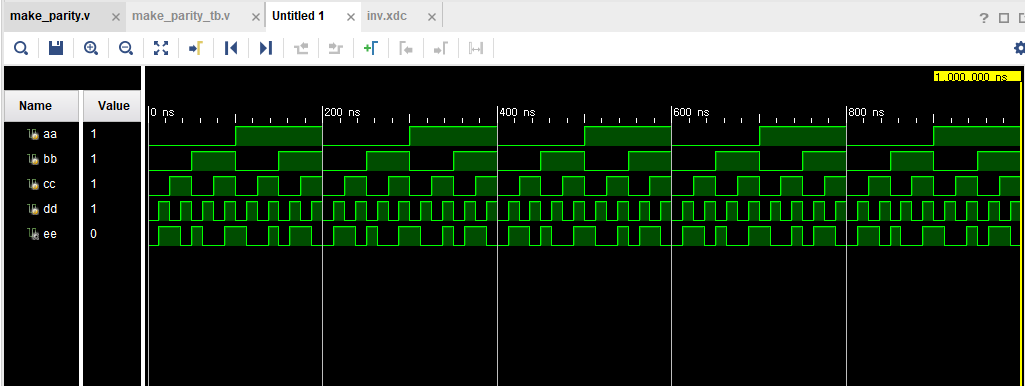
결과 및 과정:

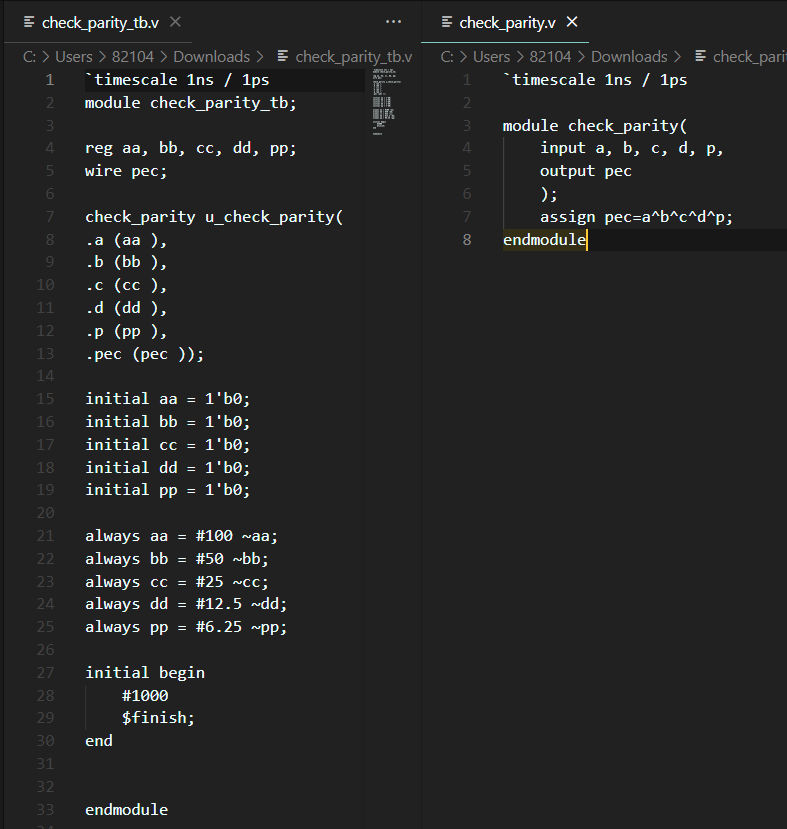
4개의 입력값이 주어지고 결과값은 1개가 나온다. 짝수 패러티를 만들어야 하기 때문에 1의 숫자가 홀수 개인지 짝수 개인지 확인해야 한다. 그래서 주어진 입력 4개에 대해서 모두 xor연산을 하고 만약 1의 개수가 홀수 개인 경우 xor연산의 결과값이 1이라서 e가 1이 된다. 그리고 1의 개수가 짝수 개이면 결과값 e가 0이 된다.

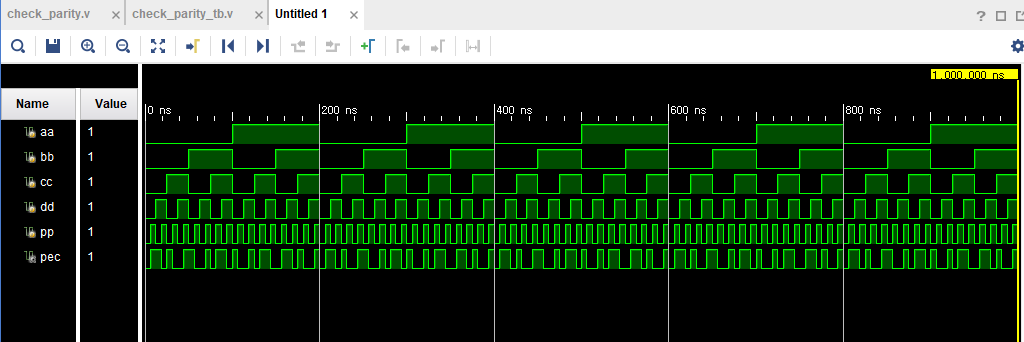
패리티 체크과정에서는 5개의 입력이 주어지는데 이는 짝수 패러티 생성기를 포함하기 때문이다. pec가 결과값으로 1개 나오는데 만약 오류가 있는 경우 즉, 5개의 입력값 중에서 1의 개수가 홀수 개인 경우는 오류라서 pec가 1이 되고 1의 개수가 짝수 개인 경우는 pec가 0이 된다.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| a | b | c | d | e |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 |

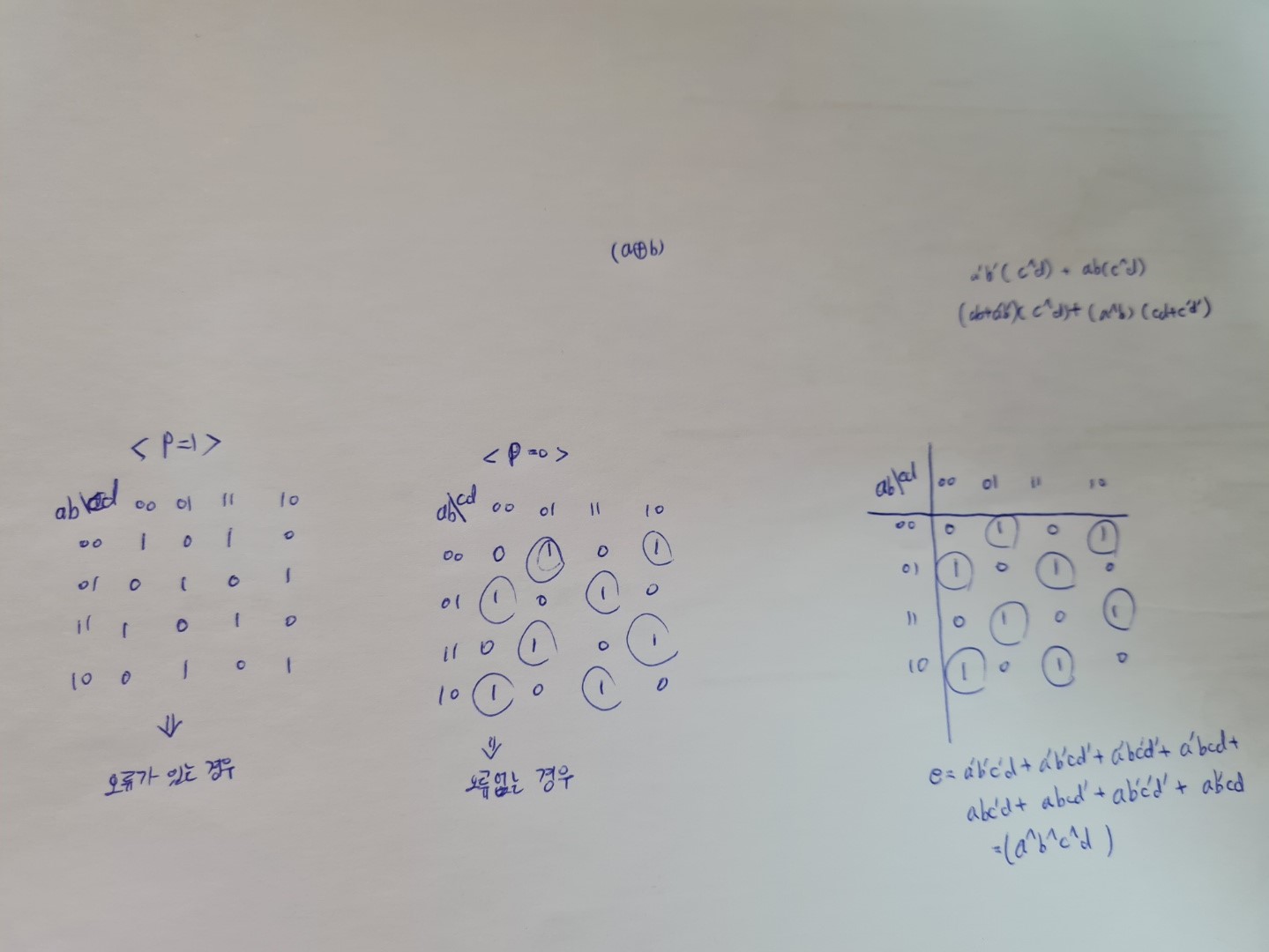








|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| A | B | C | D | P | PEC |
| 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 | 0 |
| 1 | 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 |

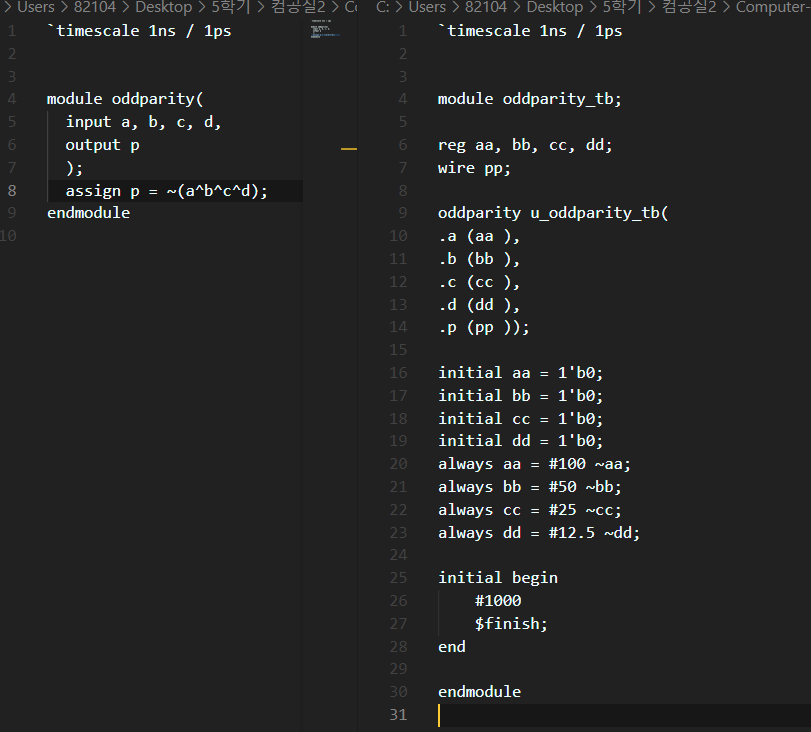


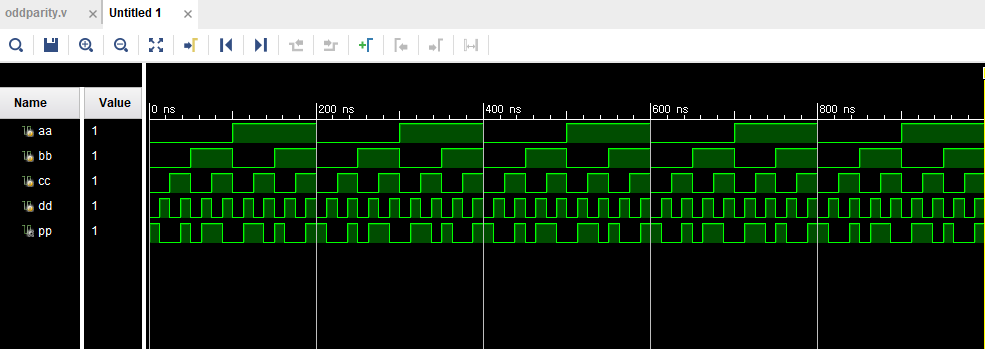
2.

결과 및 과정:

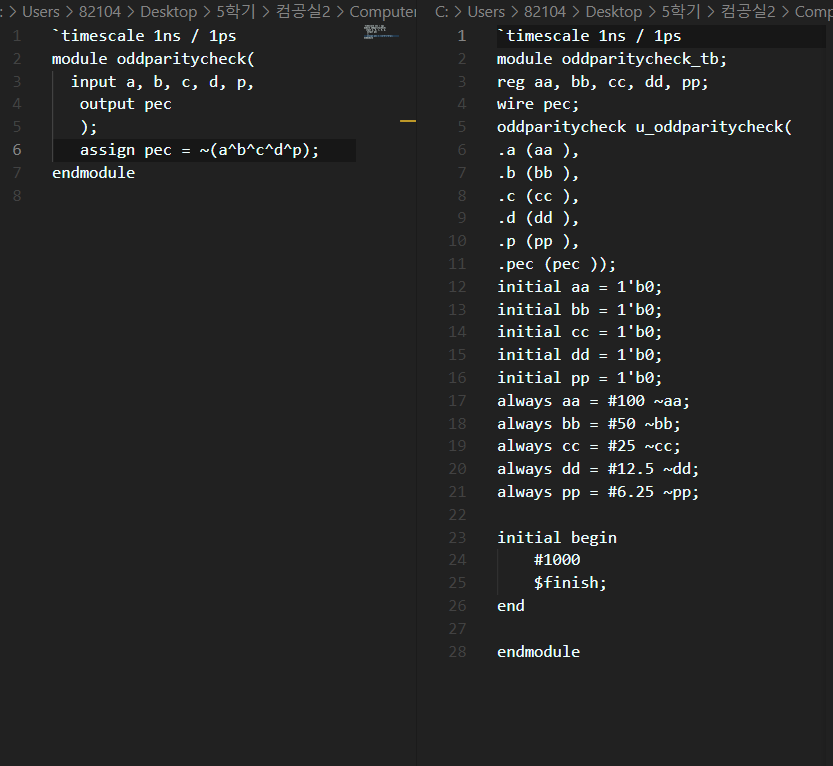
odd parity는 전체 받은 데이터에서 1의 개수가 홀수가 되도록 parity 생성자 p를 만들어주면 된다. 그래서 짝수 패러티 식에서 앞에 ~를 붙여서 부정을 해주면 된다. 입력으로 4개의 값 a,b,c,d가 들어오고 결과값으로 p가 도출된다. 이진법의 숫자들이라서 총 경우의 수가 2^4가지 즉 , 16가지가 나오게 된다.

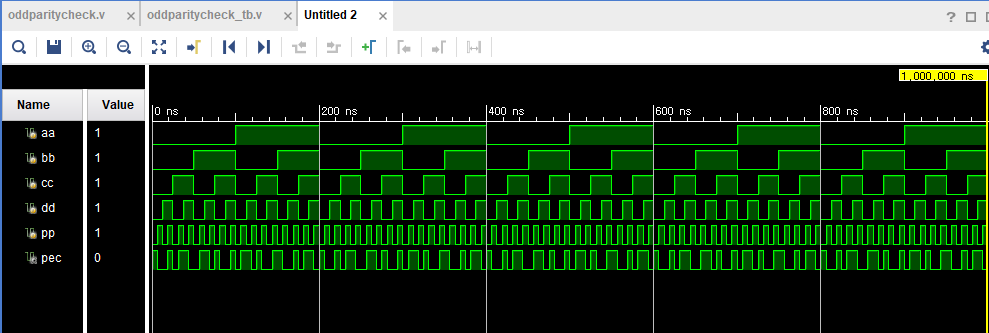
odd parity checker는 pec라는 변수 이름을 설정했고 이는 5개의 입력 a,b,c,d,p에 대해서 1의 개수가 홀수 개인 경우에 오류가 없다는 0의 값을 갖게 되고 만약 짝수 개인 경우에는 오류가 발생했다는 1의 값을 갖게 된다.



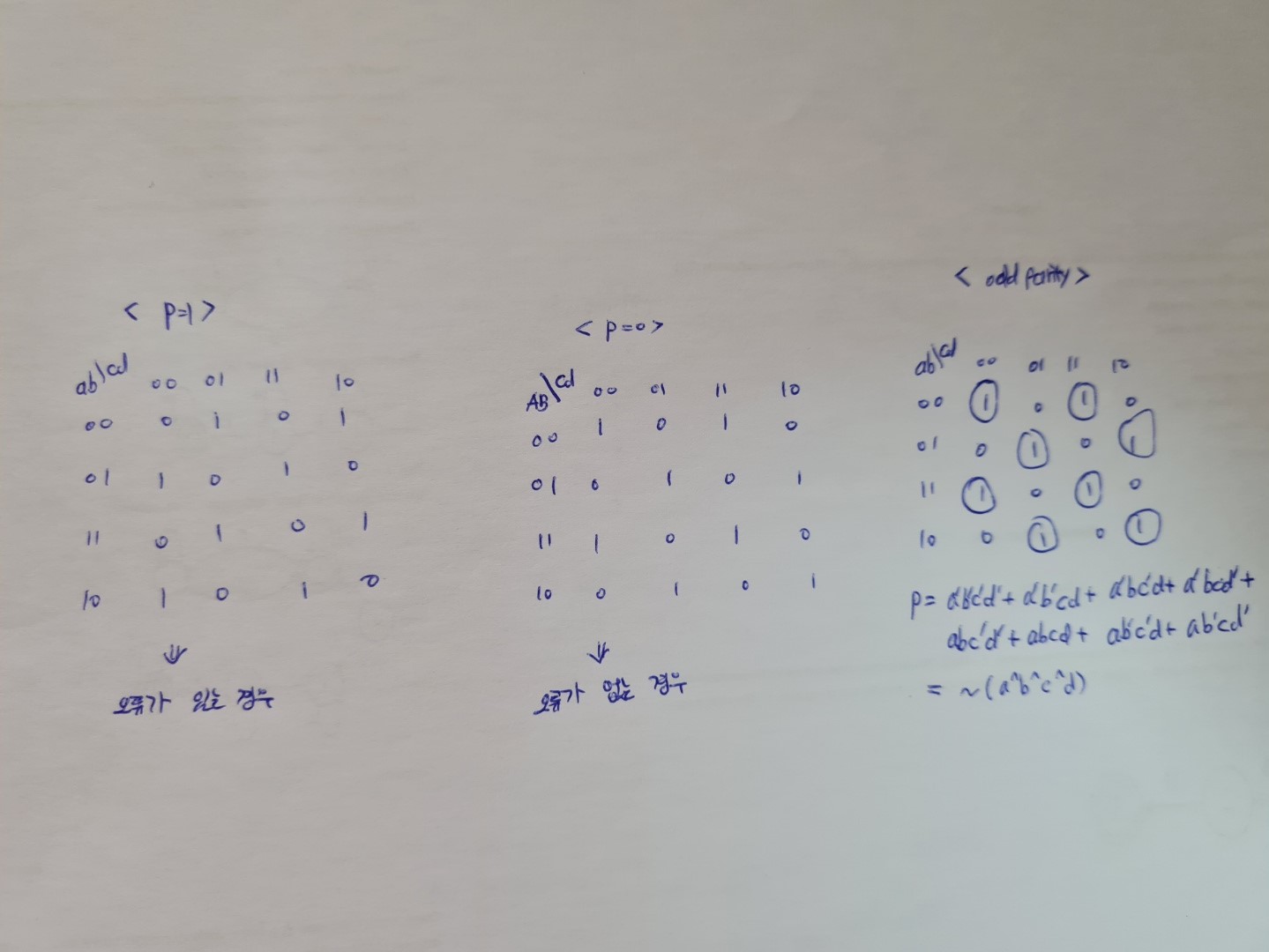


|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| a | b | c | d | p |
| 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 |





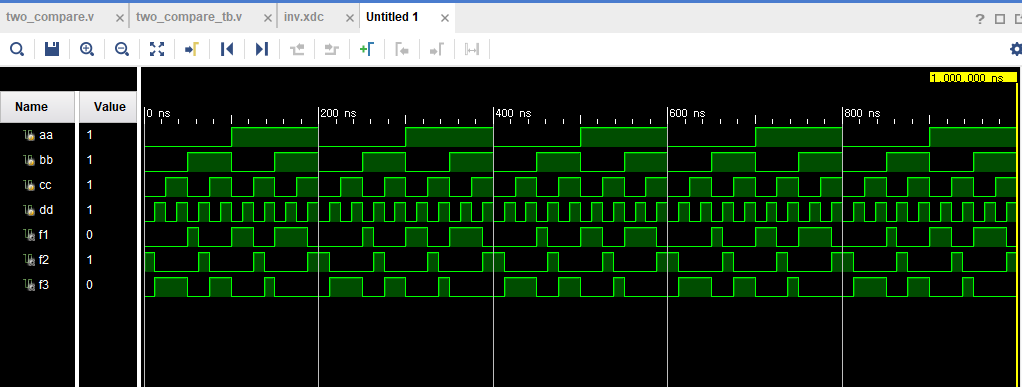
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| a | b | c | d | p | pec |
| 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 | 0 |
| 1 | 0 | 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 1 | 1 |
| 0 | 0 | 1 | 0 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 | 0 | 1 |

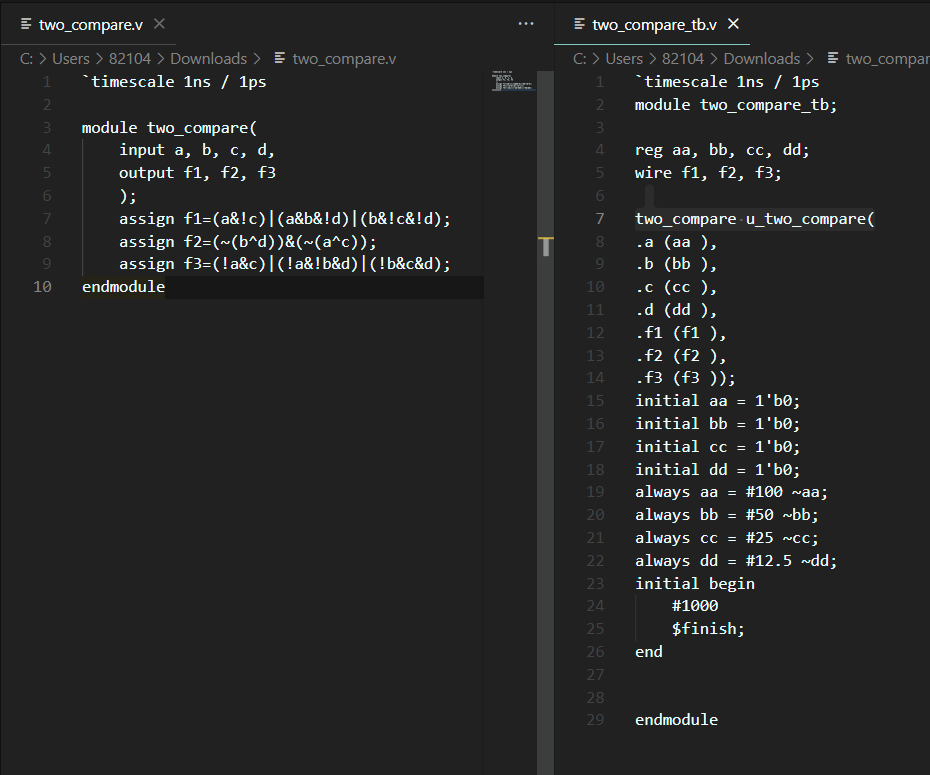


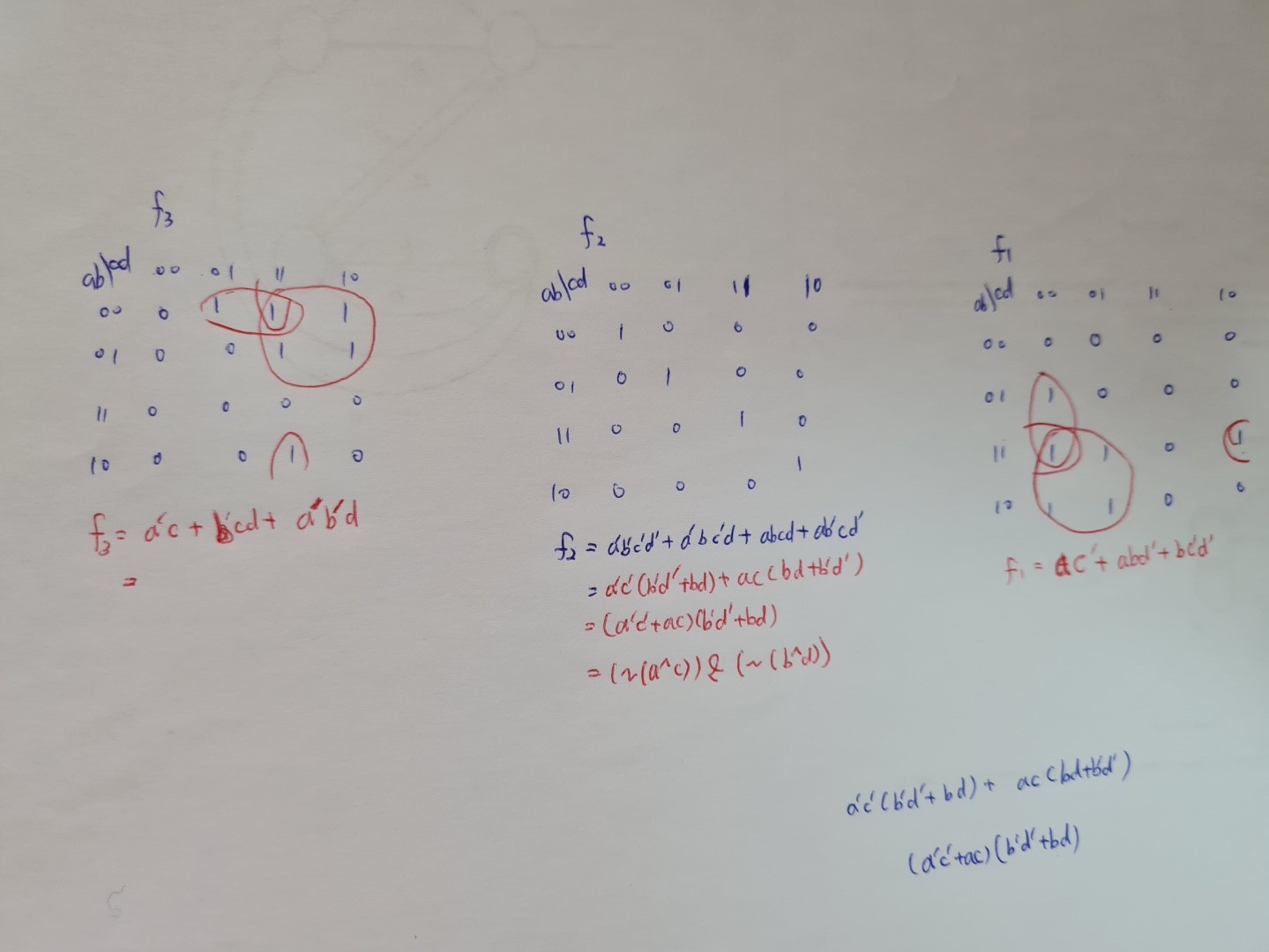
3.

결과와 과정 : a,b,c,d의 입력을 받고 결과값으로 f1,f2,f3가 출력된다. 첫 숫자가 ab, 두 번째 숫자가 cd가 된다. 그래서 만약 첫 번째 수가 두 번째 숫자보다 큰 경우에 f1이 1이 되고 두 수가 같은 경우에는 f2가 1이 된다. 그리고 두 번째 수가 더 큰 경우에는 f3가 1이 된다. 이진법의 숫자로 표현되는 두 수를 비교하기 때문에 경우의 수는 총 16가지가 되고 크기의 비교라는 특성 때문에 3개의 결과값 중에서 단 하나의 결과값만 1이 되고 나머지 두 개는 0이 된다.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| a | b | c | d | f1 | f2 | f3 |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 | 1 | 0 |







4.

카르노 맵을 통해서 논리식을 구하는 것이 생각보다 어려웠다. parity의 경우는 꼭 식의 간소화를 하지 아니해도 직관적으로 생각해 보았을 때 1의 개수가 홀수 개인지 짝수 개인지 확인만 해보면 되기 때문에 비교적 쉽게 구할 수 있다. 그런데 binary comparator를 구할 경우에는 하나씩 카르노맵을 그려서 식을 간소화 해야 하는 것 같다.

5.

4비트 크기 비교기

4비트 크기 비교기 중에서 상용화 되어 사용되는 TTL 74LS85 또는 CMOS 4063 모델들이 있고 여기에 추가적인 입력 단자가 있어서 많은 개별 비교기를 갖고 4비트 이상의 단어에 대한 크기 비교가 가능하다.

