5주차 결과보고서

전공: 경영학과 학년: 3학년 학번: 20190963 이름: 한다현

**1.**

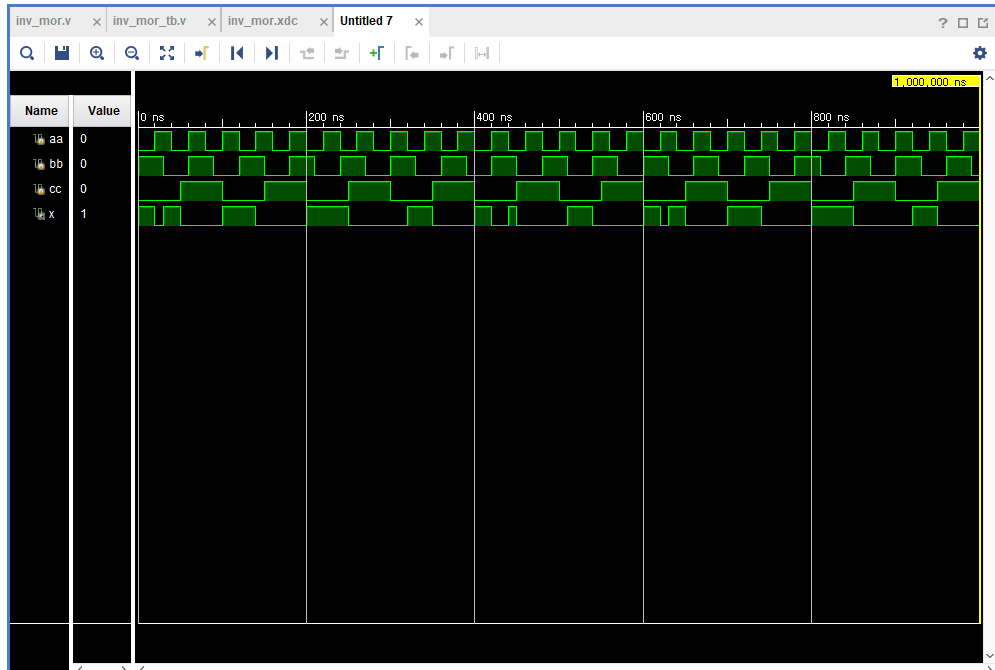
5주차 실험의 목적은 De Morgan의 정리와 Boolean 함수의 동작을 이해하고 확인하는 것이다. 먼저 Verilog를 사용하여 De Morgan의 정리와 Boolean 함수를 설계한다. De Morgan의 정리의 경우 다른 방식으로 표현된 두 식이 동일한 결과를 출력하는지 확인하는 방식으로 진행한다. Boolean 함수의 경우 여러 개의 게이트와 그에 따른 출력을 설계하고 입력이 각각의 게이트를 통해 올바른 결과를 출력하는지 확인한다. 설계가 끝나면 schematic을 통해 설계가 의도한 방식대로 됐는지 확인하고 simulation을 통해 구현된 결과를 확인한다. 마지막으로 FPGA를 통해 Verilog로 구현된 회로가 동작하는 것을 확인한다.

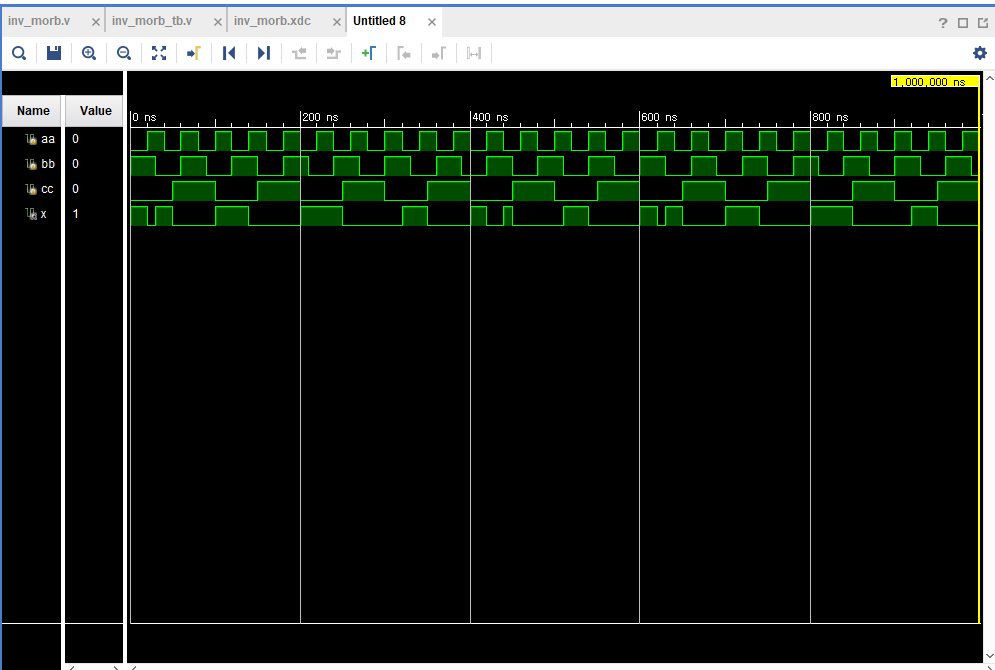
**2.**

De Morgan의 제1법칙은 A와 B의 논리합의 부정의 결과 값이 A의 부정과 B의 부정의 논리곱의 결과 값과 동일하다는 법칙이다. 즉, 입력 A와 B에 각각 부정을 취한 후 이를 AND 게이트에 입력한 출력이 A, B를 NOR 게이트에 입력한 출력과 동일하다. 시뮬레이션 결과 A와 B가 NOR 게이트에 입력된 경우에 두 입력 모두 0 일 때만 1이라는 결과를 얻었고 나머지는 모두 0이라는 결과를 얻었다. A와 B가 모두 0이면 OR 게이트를 통해 0이라는 결과를 얻고 이 결과가 인버터를 통해 1이 출력되는 과정을 거치기 때문이다. 이와 동일하게 A와 B의 각각의 부정이 AND 게이트에 입력된 경우에도 두 입력 모두 0일 때만 1이라는 결과를 얻고 나머지 경우에는 0이라는 결과를 얻었다. A와 B가 모두 0이면 A와 B의 부정은 모두 1이고 이를 AND 게이트에 입력하면 1이라는 결과가 출력되는 과정을 거치기 때문이다. De Morgan의 제2 법칙은 두 입력 A와 B의 논리곱의 부정의 출력과 A의 부정, B의 부정의 논리합의 출력이 동일하다는 법칙이다. 즉, A와 B의 부정이 OR 게이트에 입력되어 출력되는 결과와 A와 B를 NAND 게이트에 입력하여 출력되는 결과가 동일하다는 뜻이다. 시뮬레이션 결과 A와 B를 NAND 게이트에 입력하면 A와 B가 모두 1일 때만 0이 출력되고 나머지 경우에는 1이 출력된다. A와 B가 모두 1이면 AND 게이트의 결과로 1이 출력되고 이 값이 인버터를 통해 0으로 출력되기 때문이다. 마찬가지로, A와 B에 부정을 취한 뒤 OR 게이트에 입력하면 A와 B가 모두 1인 경우에만 0이 출력되고 나머지 경우에는 1이 출력된다. A와 B가 1이면 A와 B의 부정은 0이 되고 이 값이 OR 게이트에 입력되면 0이 출력되기 때문이다.

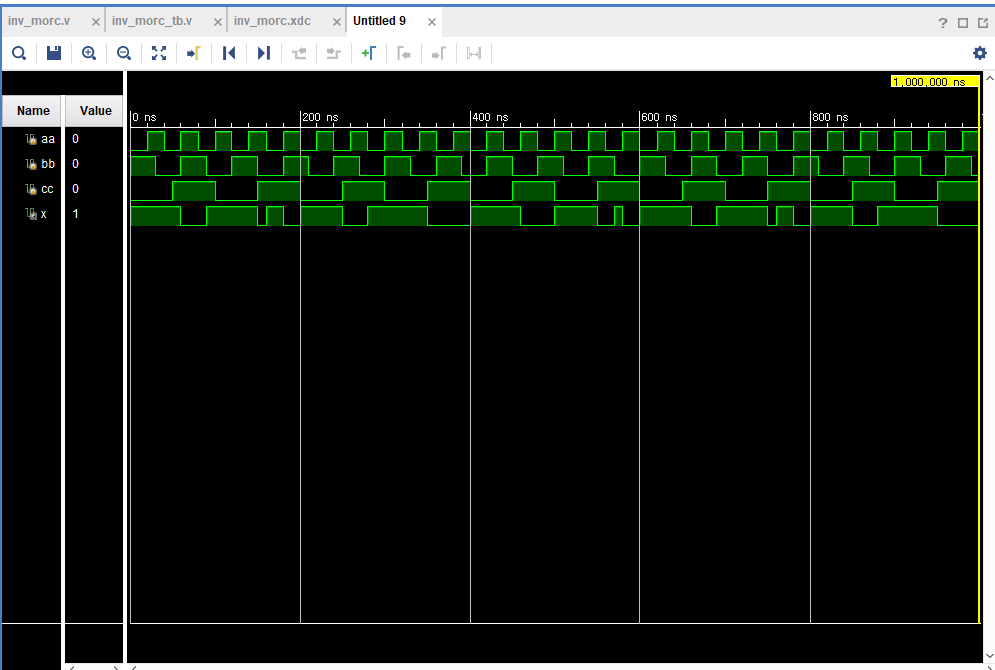
**3.**

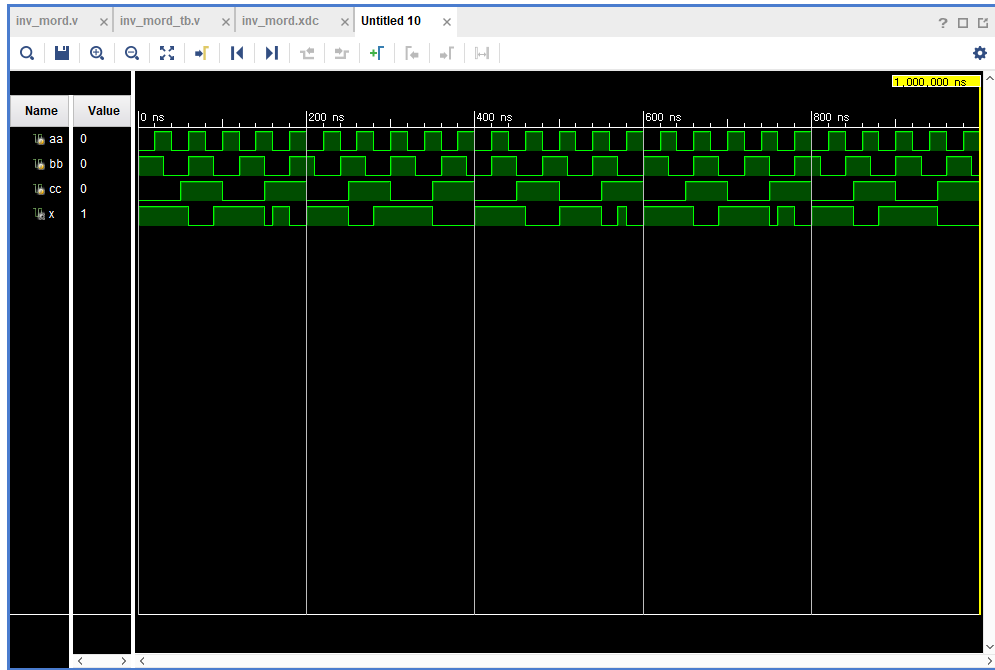
세 입력 A, B, C가 있을 때 (A’+B’)\*C’의 출력은 ((A\*B)+C)’의 출력과 동일하다. 아래의 사진 중 첫 번째 사진은 (A’+B’)\*C’의 시뮬레이션 결과이고 두 번째 사진은 ((A\*B)+C)’의 시뮬레이션 결과이다. 두 시뮬레이션은 입력 A, B, C의 변화에 따라 동일한 출력 값을 갖는 것을 볼 수 있다. A와 C의 값이 0이고 B가 1인 경우를 예로 들면 (A’+B’)\*C’에서는 A’ = 1, B’ = 0이므로 A’+B’ = 1이 되고 C’의 값은 1이기 때문에 1이 출력된다. ((A\*B)+C)’에서는 A\*B의 값은 0이고 0과 C의 OR 게이트 출력 값은 0이기 때문에 0의 부정인 1이 출력되어 (A’+B’)\*C’와 동일한 값을 가진다.





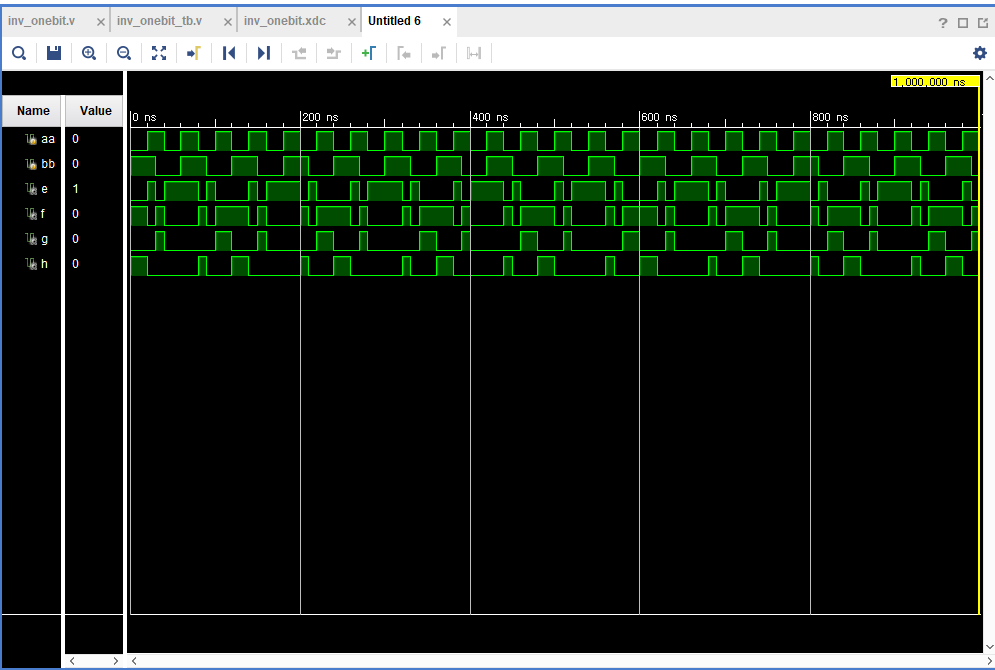
비슷하게, (A’\*B’)+C’의 출력과 ((A+B)\*C)’의 출력은 동일하다. 아래의 사진 중 첫 번째 사진은 (A’\*B’)+C’의 시뮬레이션 결과이고 두 번째 사진은 ((A+B)\*C)’의 시뮬레이션 결과이다. 두 시뮬레이션은 입력 A, B, C의 변화에 따라 동일한 출력 값을 갖는 것을 볼 수 있다. A와 C가 1이고 B가 0인 상황을 예로 들면, (A’\*B’)+C’에서는 A’ = 0, B’ = 1이므로 A’\*B’의 값은 0이고 C’은 0이다. 두 A’\*B’의 출력과 C’의 값을 OR 게이트를 통해 출력하면 0이 출력된다. ((A+B)\*C)’에서는 A+B = 1이므로 ((A+B)\*C) = 1이다. 따라서 ((A+B)\*C)’의 출력은 0으로 (A’\*B’)+C’의 출력과 동일하다.





**4.**

두 입력 A, B가 있을 때 A와 B가 입력인 XOR 게이트의 출력의 부정을 E라고 하고 A와 B가 입력인 XOR 게이트의 출력을 F라고 하고 A와 B의 부정이 입력인 AND 게이트를 G라고 하고 A의 부정과 B가 입력인 AND 게이트를 H라고 한다. 이 때 각각의 출력의 시뮬레이션 결과는 아래 사진과 같고 A, B가 입력이고 E, F, G, H가 출력인 Boolean 함수의 진리표는 다음과 같다.



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| In A | In B | Out E | Out F | Out G | Out H |
| 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 |

**5.**

De Morgan의 제1 법칙인 (A+B)’ = A’\*B’와 제2 법칙인 (A\*B)’ = A’+B’을 각각 NOR 게이트와 NAND 게이트와 비교하여 검토하였다. 이를 통해 A+B의 보수는 A와 B에 보수를 취하는 것뿐만 아니라 논리합을 논리곱으로 바꾸어 계산한 것과 같다는 결과를 얻었다. 마찬가지로 A\*B의 보수는 A와 B에 각각 보수를 취하는 것에 더해 논리곱을 논리합으로 바꾸어 계산한 것과 같다는 결과를 얻을 수 있었다. 또한 (A’+B’)\*C’ = ((A\*B)+C)’와 (A’\*B’)+C’ = ((A+B)\*C)’을 각각의 시뮬레이션을 통해 입력 A, B, C의 변화에 따라 (A’+B’)\*C’ = ((A\*B)+C)’가 항상 동일한 값을 출력하고 (A’\*B’)+C’ = ((A+B)\*C)’가 항상 동일한 값을 출력한다는 결론을 얻을 수 있었다. 마지막으로 두 개의 입력이 여러 게이트의 입력이 되어 계산될 수 있다는 것을 Boolean 함수의 결과를 통해 얻을 수 있었다. 위에 제시된 네 개의 게이트만이 아니라 다른 여러 종류의 게이트도 Boolean 함수로 사용될 수 있고 4개보다 더 많은 수의 게이트를 사용할 수도 있다. 추가적인 논의 사항으로는 (A’+B’)\*C’ = ((A\*B)+C)’와 (A’\*B’)+C’ = ((A+B)\*C)’을 제외하고 드모르간의 법칙으로 검증할 수 있는 식이 더 있는지, 2개의 입력으로 얼마나 많은 게이트를 한번에 처리할 수 있는지 등을 고려할 수 있다.

**6. 추가 이론 조사 및 작성**

부울 함수에는 minterm과 maxterm일나느 개념이 있다. Minterm과 maxterm을 이해하기 위해서는 먼저 product term과 sum term을 이해해야 하는데 product term(곱항)은 한 개 이상의 변수가 AND로 연결된 항을 의미하고 sum term(합항)은 한 개 이상의 변수가 OR로 연결된 항을 의미한다. Minterm은 모든 변수가 항상 한 번씩 사용된 곱항이고 maxterm은 모든 변수가 한 번씩 사용된 합항이다. 즉 변수 x, y, z가 있을 때, minterm은 xyz, x’yz’ 등의 항을 의미하는 것이고 maxterm은 x+y+z, x’+y+z 등의 항을 의미하는 것이다. Minterm의 기본 값은 1이고 maxterm의 기본 값은 0이다.