5주차 결과보고서

전공 : 국제한국학과 학년 : 4학년 학번 : 20181202 이름 : 김수미

**1. 실험 목적**

- 드모르간의 정리와 Boolean 함수의 동작을 이해하고 직접 논리식을 작성해본다.

- Verilog 언어를 사용해 드모르간의 정리 및 Boolean function의 동작을 구현한다.

- Vivado 에서 입력 신호를 생성하여 Simulation을 통해 구현된 결과를 확인한다.

- FPGA를 통해 Verilog로 구현된 회로의 동작을 확인한다.(강의자료에는 실험 목표로써 기재되어 있으나 코로나 비대면 수업으로 인해 FPGA 실습은 진행하지 않았다).

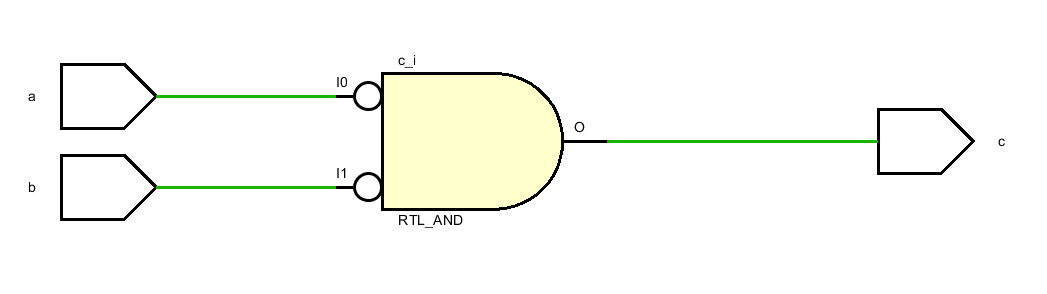
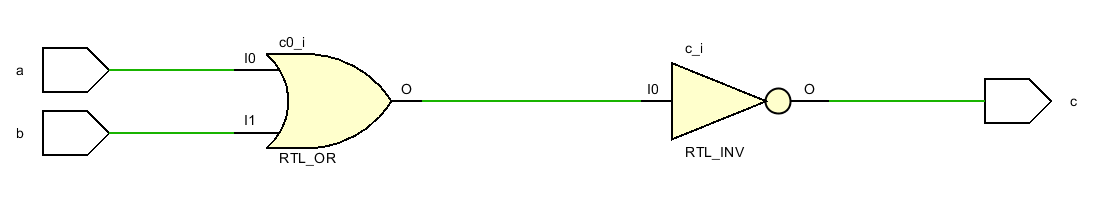
**2. De-Morgan 의 제 1,2 법칙의 simulation 결과 및 과정에 대해서 설명하시오.**

**1) Truth Table**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **A** | **B** | **A+B** | **A•B** | **A’** | **B’** | **(A+B)’** | **A’•B’** | **(A•B)’** | **A’+B’** |
| 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

**2) 드모르간의 제 1 법칙 :**

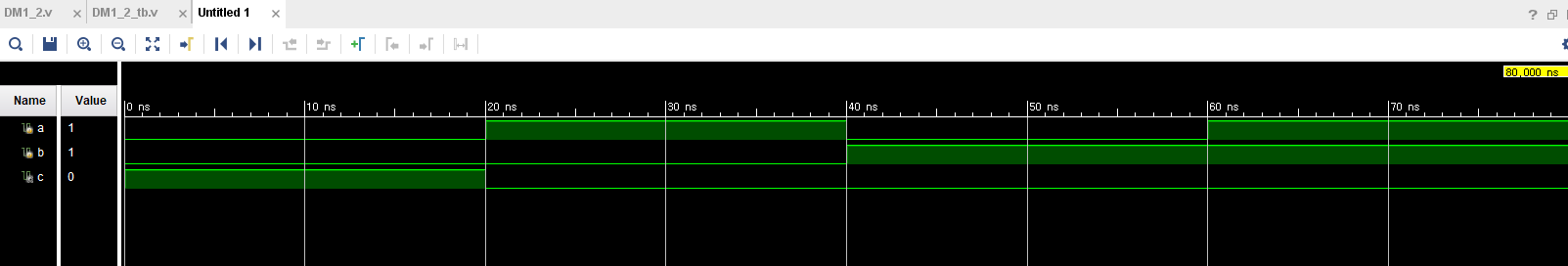
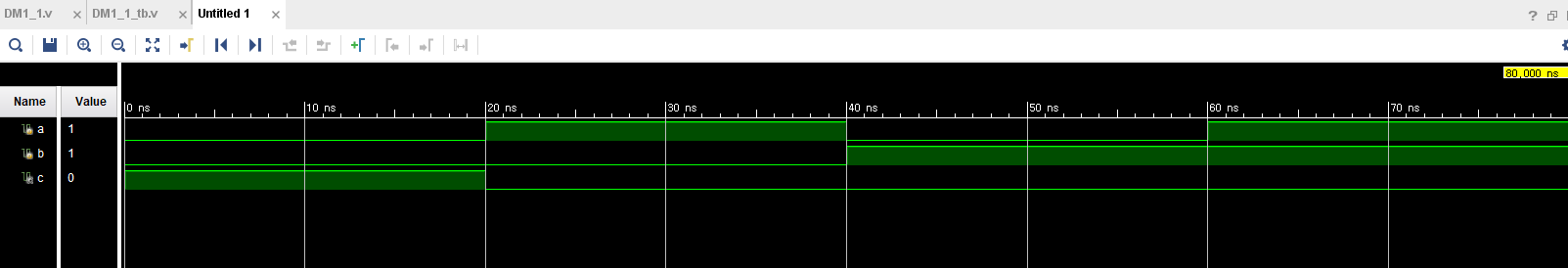
**1) Schematic**

****

**2) Source Code**

|  |  |  |
| --- | --- | --- |
| **(A)’s Design Source** | **(A)’s Test Bench** | **(B)’s Test Bench** |
|  |  |  |
| **(B)’s Design Source** |
|  |

**3) Simulation**

****

**4) NOR와 비교**

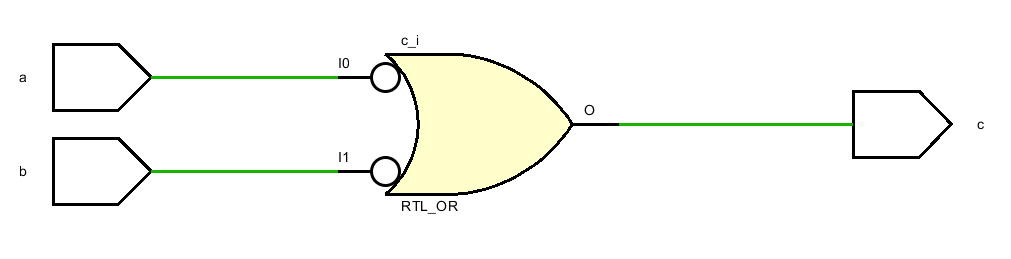
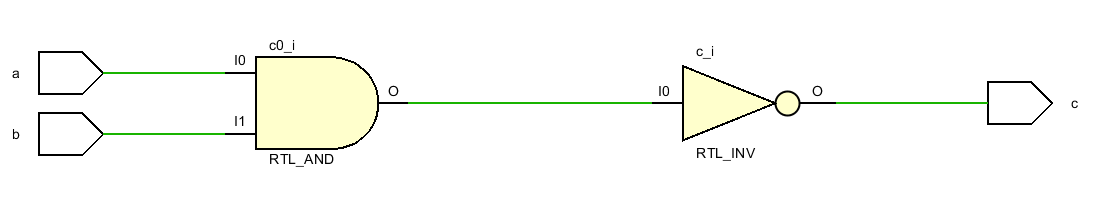
4주차에 살펴본 NOR 게이트는 OR 게이트에 NOT 게이트를 연결시킨 것으로, (A)의 형식과 정확히 동일하다. 즉 A NOR B 는 ~(A OR B) = 이므로 드모르간 제 1법칙에서 살펴보고 있는 (A), (B), (A NOR B) 세가지 식은 모두 동일하다.

**5) 결과 및 과정 분석**

Simulation 결과 및 Truth Table을 보면 두 논리식 의 연산 결과는 동일한 것을 확인할 수 있다. 즉 두 표현은 같은 연산 결과를 반환하므로, 복잡한 논리식을 간소화 할 때 이 성질을 이용하면 용이한 경우가 많다.

**3) 드모르간의 제 2 법칙 :**

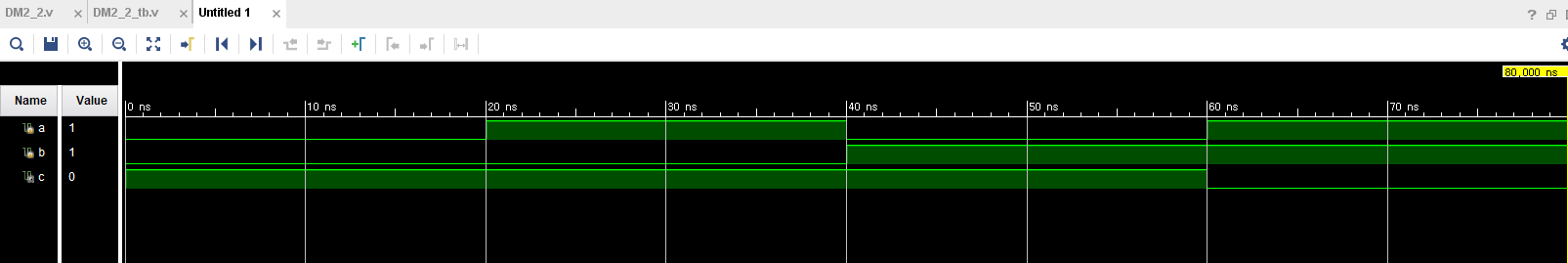
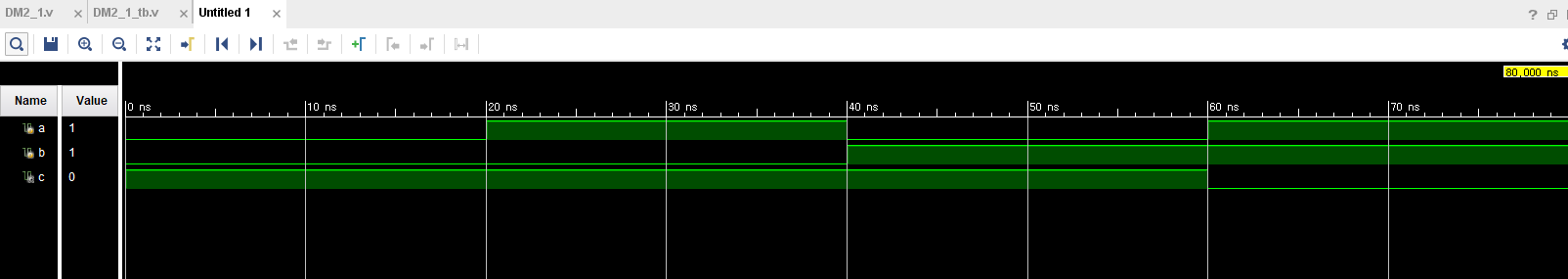
**1) Schematic**

****

**2) Source Code (Design Source / Test Bench)**

|  |  |  |
| --- | --- | --- |
| **(A)’s Design Source** | **(A)’s Test Bench** | **(B)’s Test Bench** |
|  |  |  |
| **(B)’s Design Source** |
|  |

**3) Simulation**

****

**4) NAND와 비교**

4주차에 살펴본 NAND 게이트는 AND 게이트에 NOT 게이트를 연결시킨 것으로, (A)의 형식과 정확히 동일하다. 즉 A NAND B 는 ~(A AND B) = 이므로 드모르간 제 1법칙에서 살펴보고 있는 (A), (B), (A NAND B) 세가지 식은 모두 동일하다.

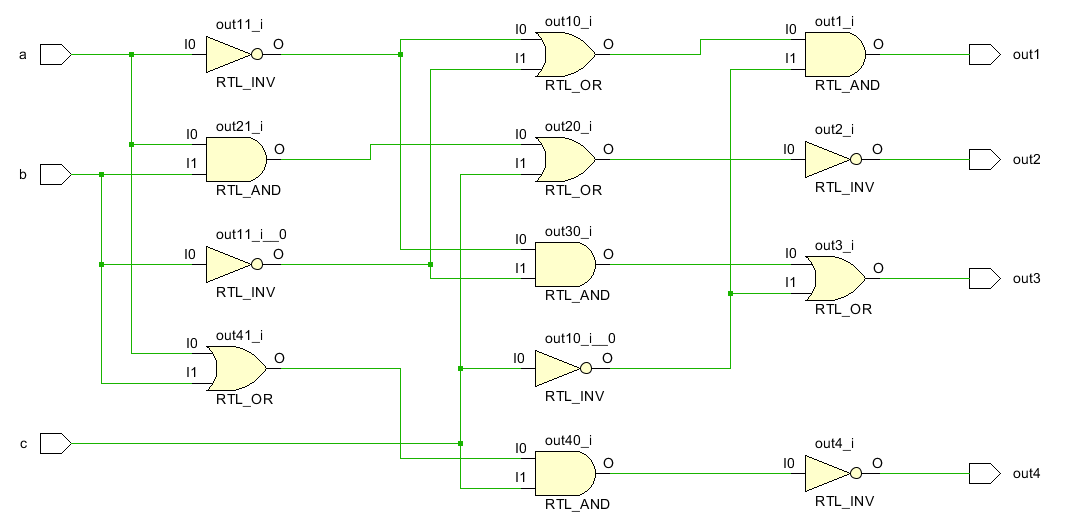
**5) 결과 및 과정 분석**

Simulation 결과 및 Truth Table을 보면 두 논리식 의 연산 결과는 동일한 것을 확인할 수 있다. 즉 두 표현은 같은 연산 결과를 반환하므로, 복잡한 논리식을 간소화 할 때 이 성질을 이용하면 용이한 경우가 많다.

**3. Boolean Function 의 simulation 결과 및 과정에 대해서 설명하시오.**

**1) (A'+B')\*C' = ((A\*B)+C)' / (A'\*B')+C' = ((A+B)\*C)'**

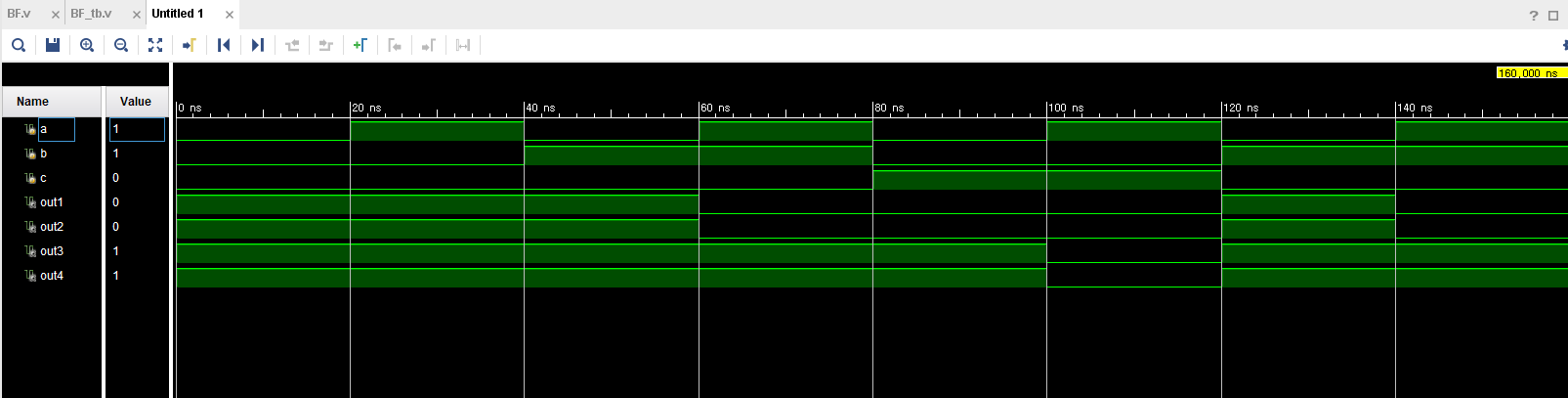
**1) Schematic**

****

**2) Source Code (Design Source / Test Bench)**

|  |  |
| --- | --- |
| **Design Source** | **Test Bench** |
|  |  |

**3) Simulation**

****

**4) Truth Table**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A** | **B** | **C** |  |  |  |
| 0 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 | 0 |
| **A’** | **B’** | **C’** |  | **’** | |
| 1 | 1 | 1 | 1 | 1 | |
| 0 | 1 | 1 | 1 | 1 | |
| 1 | 0 | 1 | 1 | 1 | |
| 1 | 1 | 0 | 1 | 0 | |
| 0 | 0 | 1 | 0 | 0 | |
| 0 | 1 | 0 | 1 | 0 | |
| 1 | 0 | 0 | 1 | 0 | |
| 0 | 0 | 0 | 0 | 0 | |

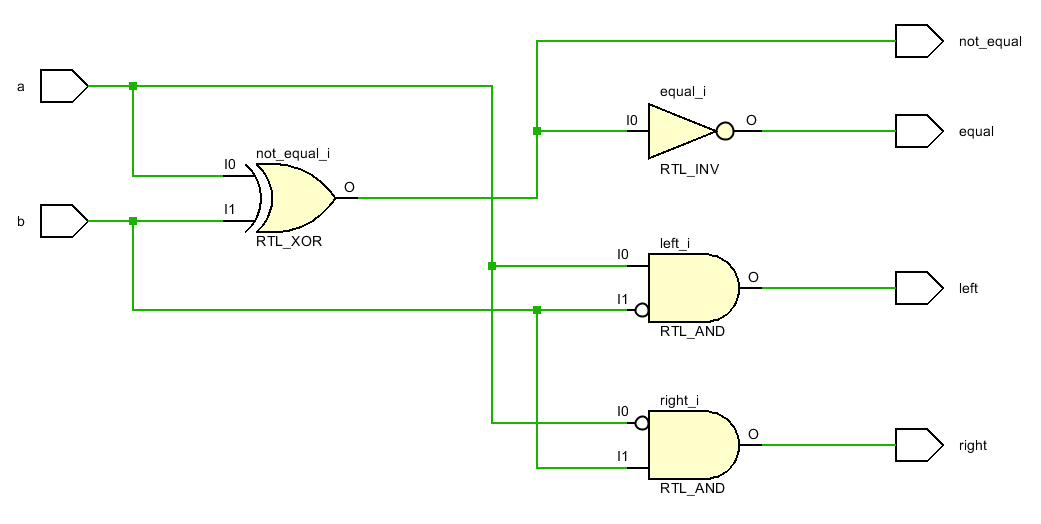
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A** | **B** | **C** |  |  |  |
| 0 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 | 0 |
| **A’** | **B’** | **C’** |  | **’** | |
| 1 | 1 | 1 | 1 | 1 | |
| 0 | 1 | 1 | 0 | 1 | |
| 1 | 0 | 1 | 0 | 1 | |
| 1 | 1 | 0 | 1 | 1 | |
| 0 | 0 | 1 | 0 | 1 | |
| 0 | 1 | 0 | 0 | 0 | |
| 1 | 0 | 0 | 0 | 0 | |
| 0 | 0 | 0 | 0 | 0 | |

**5) 결과 및 과정 분석**

Simulation 결과 및 Truth Table을 보면 두 논리식 (A'+B')\*C' = ((A\*B)+C)' 과 (A'\*B')+C' = ((A+B)\*C)'이 성립함을 확인할 수 있다. 이는 앞에서 확인한 드모르간 법칙을 이용하여 유도하는 것도 가능하다.

**4. 1Bit 비교기의 simulation 결과 및 과정에 대해서 설명하시오.**

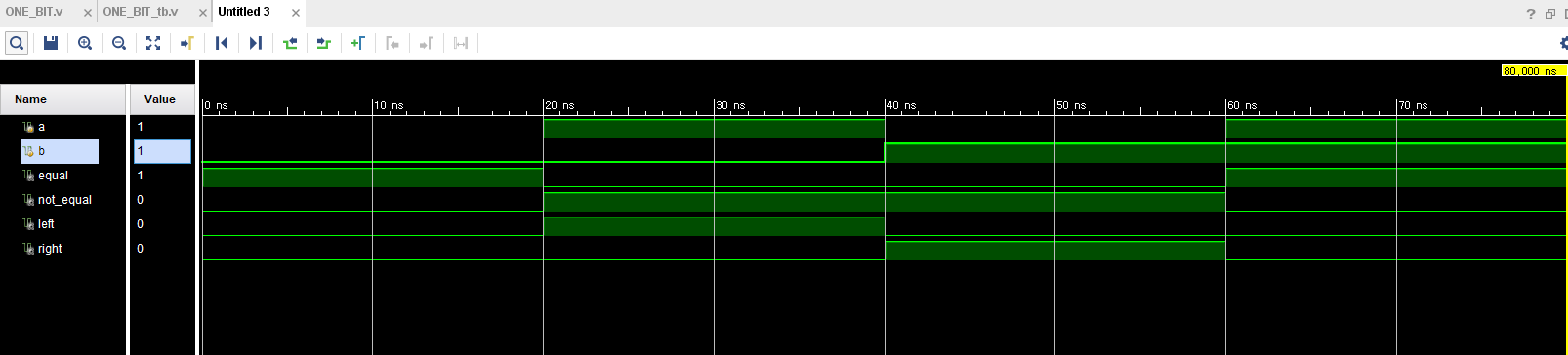
**1) Schematic**

****

**2) Source Code (Design Source / Test Bench)**

|  |  |
| --- | --- |
| **Design Source** | **Test Bench** |
|  |  |

**3) Simulation**

****

**4) Truth Table**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
| 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) 결과 및 과정 분석**

비교는 A XOR B 연산으로, 비교는 A XNOR B 연산으로,  
 연산은 연산으로, 연산은 으로 구현할 수 있다.

대소 비교 연산을 이와 같이 구현할 수 있는 이유는 input이 1 아니면 0의 값으로 주어지기 때문이다.

**5. 결과 검토 및 논의 사항**

1 Bit 비교기 구현에서 두 input의 대소 비교를 논리연산자로 구현할 수 있다는 사실을 새롭게 발견할 수 있었다. 또한 실습을 진행하면서 Simulation 결과를 확인할 때, 각 변수 값의 변화 주기(#시간)를 2의 제곱수로 두면 모든 경우의 수를 비교할 때 용이하다는 점을 새롭게 발견했다.

**6. 추가 이론 조사 및 작성**

**1) 드모르간 법칙의 증명과 활용**

실습에서는 직접 input과 output을 비교함으로써(진리표 작성) 드모르간 법칙이 성립함을 확인했다. 집합의 영역에서 드모르간 법칙을 사용하면 벤다이어그램을 이용해서도 드모르간 법칙을 증명할 수 있다.



전자회로 설계에서는 SOP와 POS 식을 올바르게 작성하고 이를 간소화 하는 것이 효율적인 회로를 설계하는데 매우 중요한데, SOP식을 POS 식으로 변환하거나, POS 식을 SOP식으로 변환할 때 드모르간 법칙을 사용하면 손쉬운 변환이 가능하다.

예 ) 다음과 같은 SOP(sum of product)식이 있다 :   
이 식의 전체에 NOT을 취하면 :

즉 SOP 식에 NOT을 취하면 드모르간 법칙에 의해 POS(product of sum) 형식이 된다.