5주차 결과보고서

전공: 컴퓨터공학과 학년: 4학년 학번: 20191583 이름: 김태곤

**1. 실험 목적**

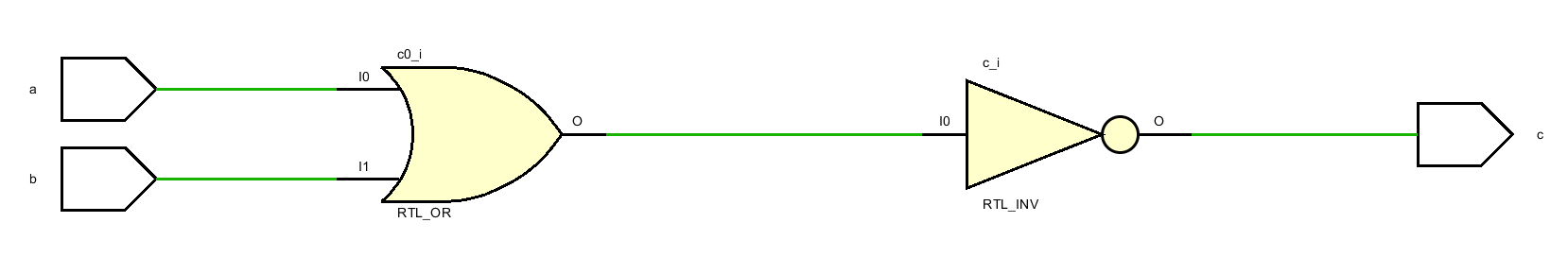
Verilog HDL을 활용하여 드모르간의 법칙, 복합 논리식의 변환, 및 1비트 비교기 설계 및 시뮬레이션을 통해 디지털 논리 설계의 핵심 개념을 깊이 이해하고 실제 FPGA를 사용한 구현을 통해 이론과 실습의 통합을 강화하는 것을 목적으로 한다. 이 과정에서, Verilog HDL을 사용하여 드모르간의 제1, 제2 법칙을 적용한 논리 회로 설계, 복합 논리식 (A'+B')\*C' = ((A\*B)+C)' 의 변환 및 구현, 그리고 간단한 1비트 비교기의 설계 및 시뮬레이션을 수행한다. 이러한 실험을 통해, 각 논리 회로의 동작 원리와 논리식 변환의 중요성을 깊이 이해하고, 설계한 회로의 정확성과 효율성을 Vivado 프로그램을 사용한 시뮬레이션과 실제 FPGA 구현을 통해 평가한다.

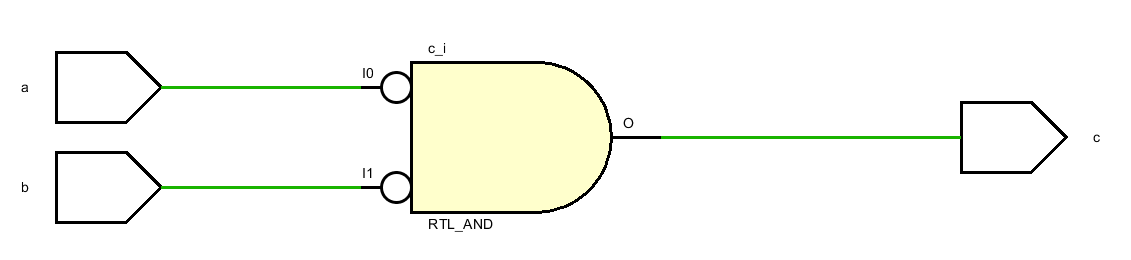
**2. De-Morgan 의 제 1,2 법칙의 simulation 결과 및 과정에 대해서 설명하시오.**

**(NAND,NOR 과 비교 포함)**

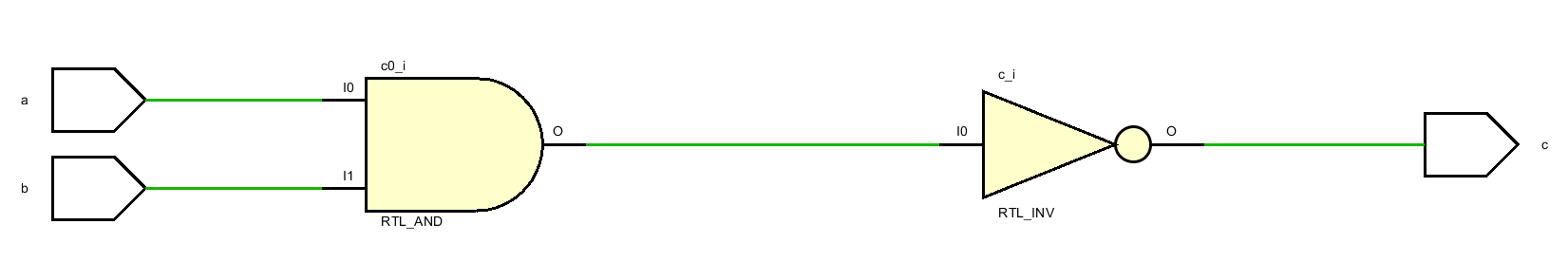
1. **Schematic**

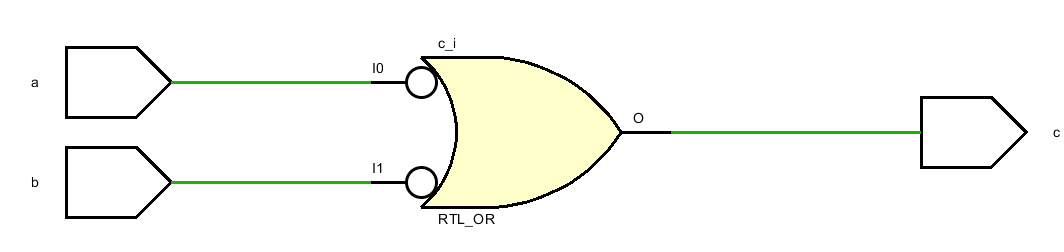
**1)) 제 1법칙**

****

****

**2)) 제 2법칙**

****

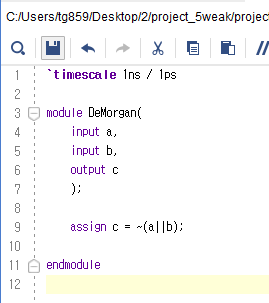
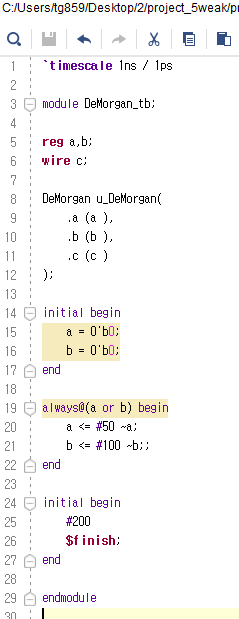
****

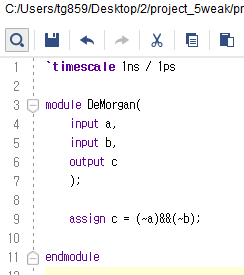
1. **Design Source와 Testbench file**

Testbench file은 모두 동일한 코드로 진행하였다.

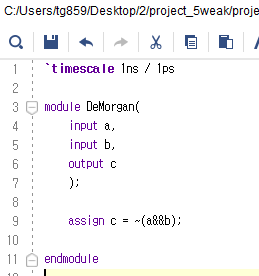
아래 코드는 각각 제1법칙과 2법칙에 해당하는 코드이다.

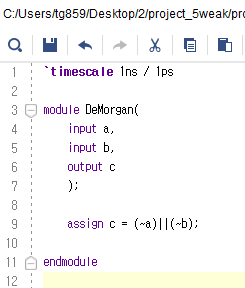
**1)) 제 1법칙**

****

****

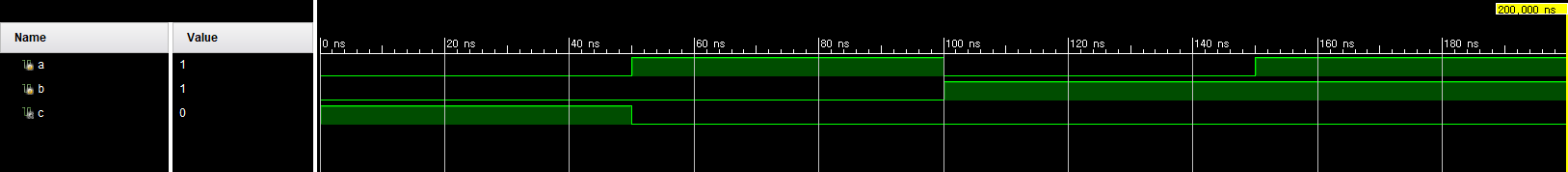
**2)) 제 2법칙**

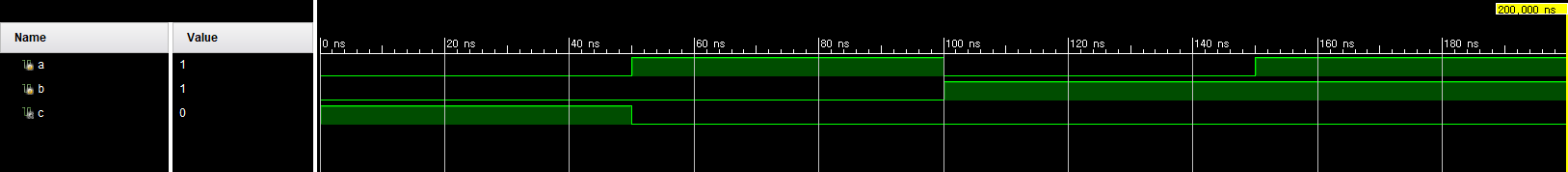
****

****

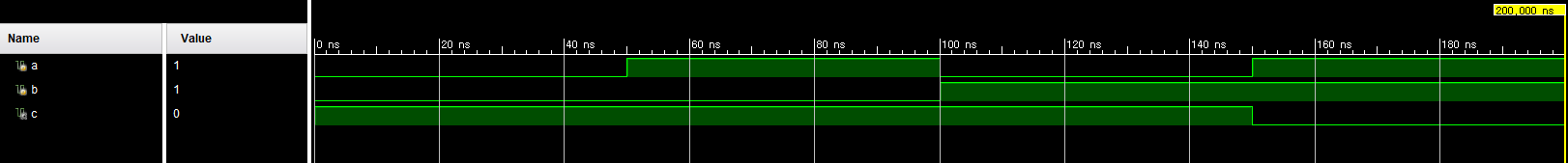
1. **Simulation 결과**

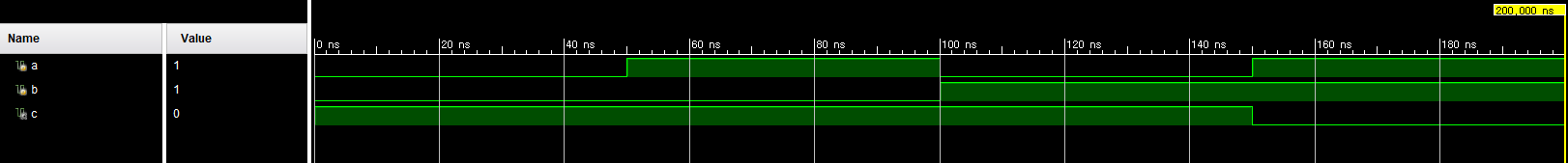
**1)) 제 1법칙**

****

****

**2)) 제 2법칙**

****

****

1. **진리표**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A(in) | B(in) | 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 |

1. **결과 및 구현 과정**

**1)) 제 1법칙**

입력 a,b,와 출력 c가 주어졌을 때, 드모르간의 제 1 법칙을 구현하기 위해 먼저 a,b에 대한 OR연산 후 NOT 연산을 적용하여 출력하였다. 또한, 변환 과정을 실험하기 위해 A의 NOT연산과 B의 NOT 연산을 수행한 후 AND연산을 통해 두 식을 비교하였다. 구현 결과 2-3)의 simulation과 같은 동일한 결과가 나왔다. a와 b 모두 0일 때 1의 값을 가지고, 그 외에는 0의 값을 가진다. 이러한 동일한 결과를 통해 (A+B)’ = (A’•B’)임을 알 수 있다.

**2)) 제 2법칙**

입력 a,b,와 출력 c가 주어졌을 때, 드모르간의 제 2 법칙을 구현하기 위해 먼저 a,b에 대한 AND연산 후 NOT 연산을 적용하여 출력하였다. 또한, 변환 과정을 실험하기 위해 A의 NOT연산과 B의 NOT 연산을 수행한 후 OR연산을 통해 두 식을 비교하였다. 구현 결과 2-3)의 simulation과 같은 동일한 결과가 나왔다. a와 b 모두 1일 때 0의 값을 가지고, 그 외에는 1의 값을 가진다. 이러한 동일한 결과를 통해 (A•B)’ = (A’+B’)임을 알 수 있다.

1. **NAND, NOR과 비교**

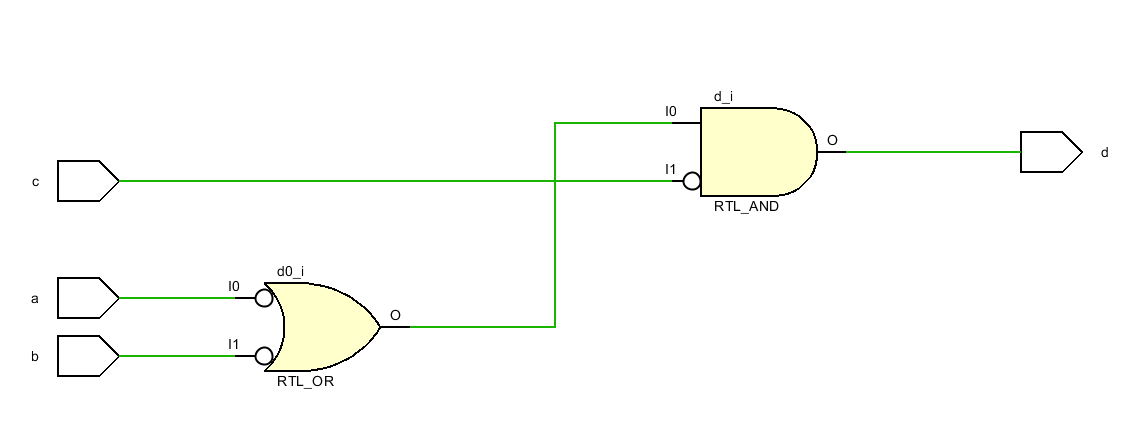
NAND 게이트는 AND 게이트에 NOT 연산을 수행하는 게이트이다. A NAND B = (A•B)’이므로 제 2법칙의 결과와 동일하다. NOR 게이트는 OR 게이트에 NOT 연산을 수행하는 게이트이다. A NOR B = (A+B)’ 이므로 제 1법칙의 결과와 동일하다. 즉 NAND와 NOR은 드모르간의 법칙에 의해 같은 연산 결과를 가지는 다른 식으로 표현할 수 있으며, 논리식을 간소화 할 때 유용하다.

**3. (A'+B')\*C' = ((A\*B)+C)' 의 simulation 결과 및 과정에 대해서 설명하시오.**

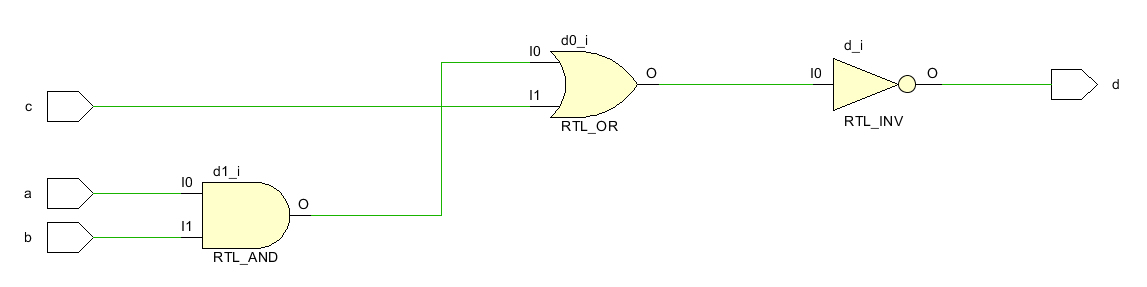
**[+ 및 \* 위치 바꾼 모양도 수행하셔야 합니다.]**

1. **Schematic**

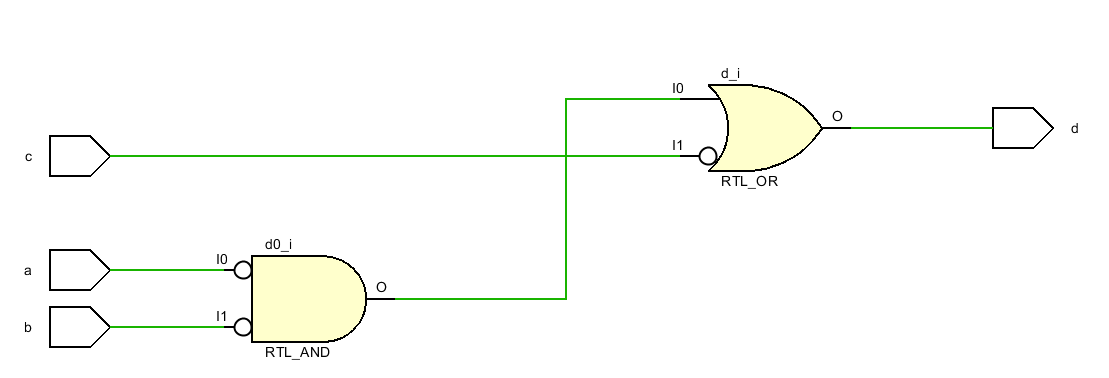
**1)) (A'+B')\*C'**



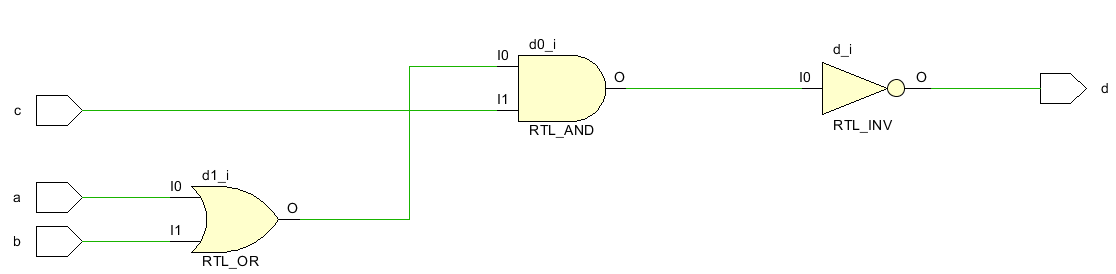
**2)) ((A\*B)+C)'**



**3)) (A'\*B')+C'**



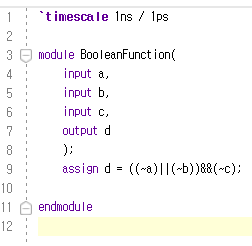
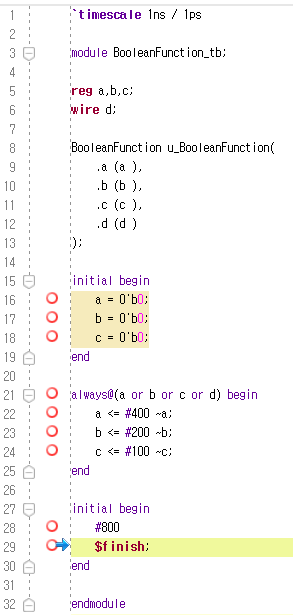
**4)) ((A+B)\*C)'**



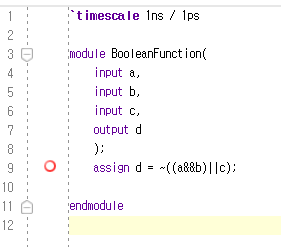
1. **Design Source와 Testbench file**

Testbench file은 모두 동일한 코드로 진행하였다.

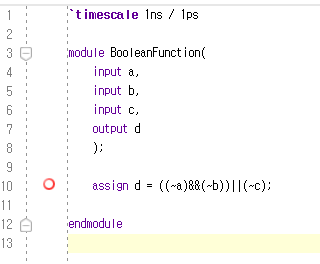
**1)) (A'+B')\*C'**



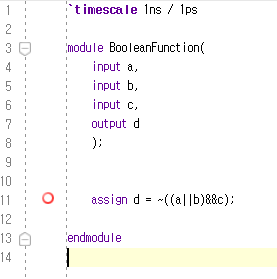
**2)) ((A\*B)+C)'**



**3)) (A'\*B')+C'**

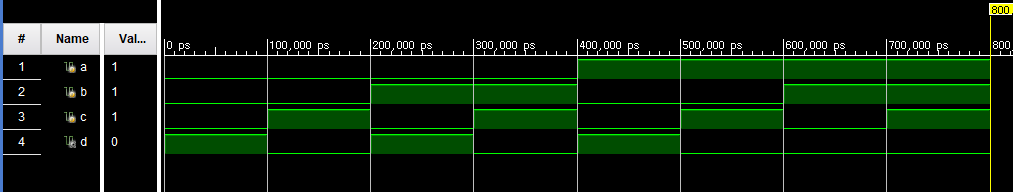


**4)) ((A+B)\*C)'**

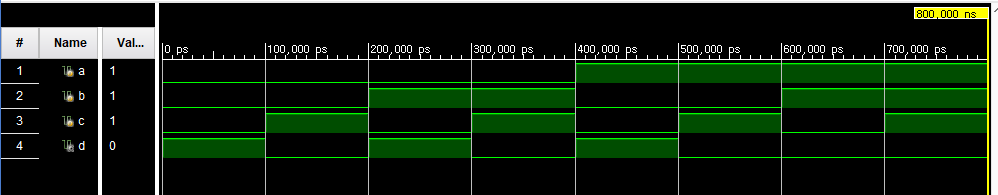


1. **Simulation 결과**

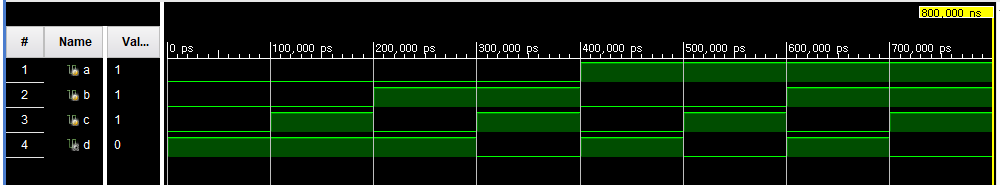
**1)) (A'+B')\*C'**



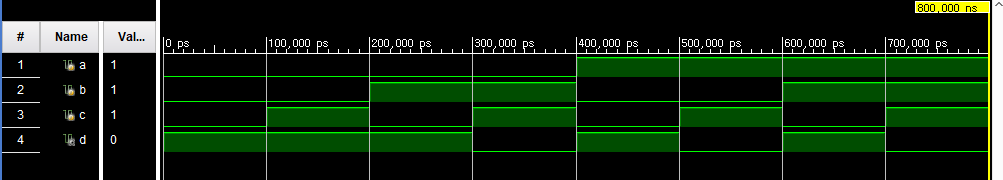
**2)) ((A\*B)+C)'**



**3)) (A'\*B')+C'**



**4)) ((A+B)\*C)'**



1. **진리표**

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

|  |  |  |  |
| --- | --- | --- | --- |
| A(in) | B(in) | C(in) | D(out) |
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 |

**2)) (A'\*B')+C' = ((A+B)\*C)'**

|  |  |  |  |
| --- | --- | --- | --- |
| A(in) | B(in) | C(in) | D(out) |
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 |

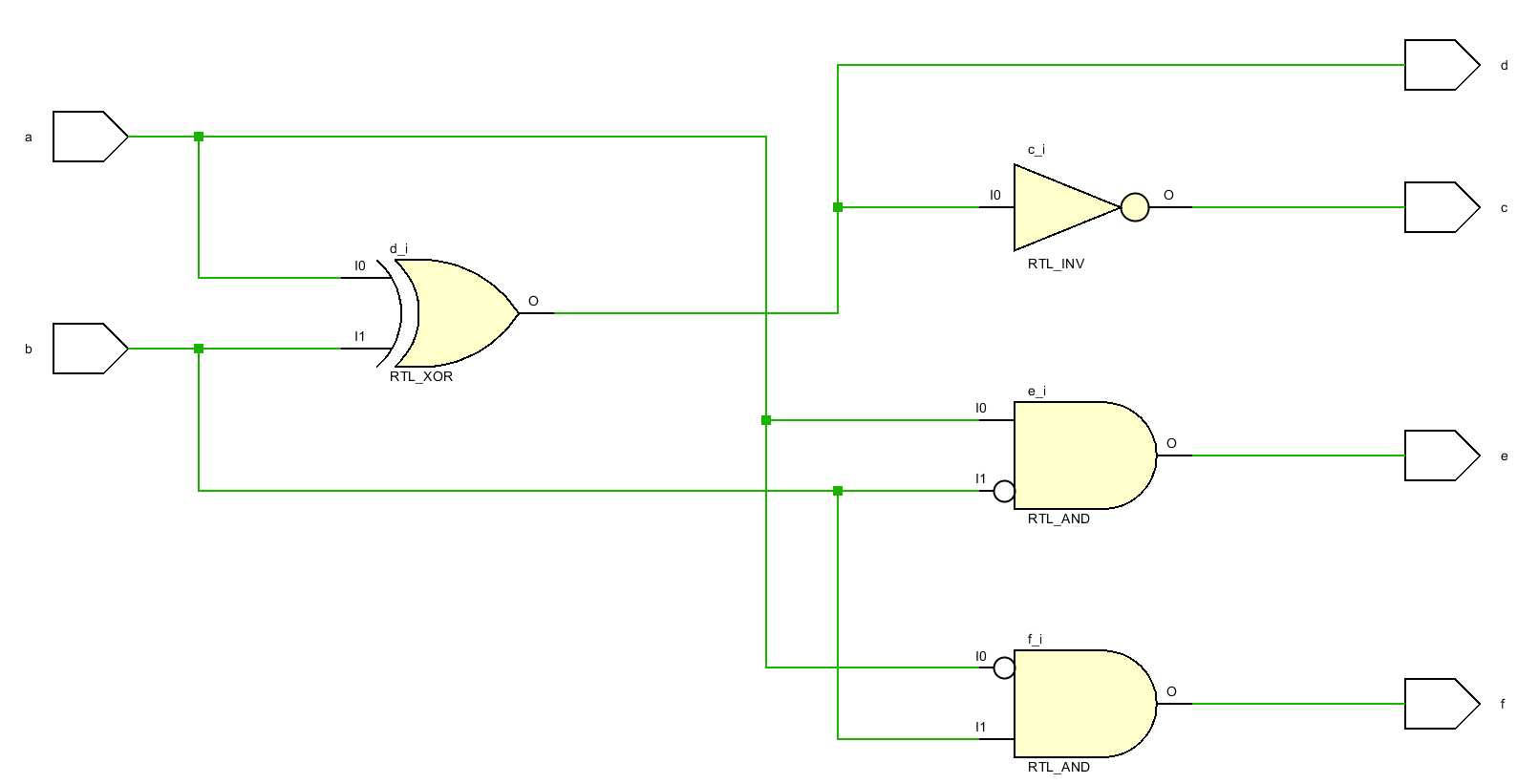
1. **결과 및 구현 과정**

입력 a,b,c와 출력 d를 이용하여, 해당 Boolean 식에 해당하는 연산을 수행하였다. 각각의 boolean식에 해당하는 연산은 3-2)의 Design source file에서 assign한 방식으로 구현하였다. 구현 결과 3-3)의 simulation과 같은 결과가 나왔다. 해당 simulation을 통해 (A'+B')\*C' = ((A\*B)+C)'와 (A'\*B')+C' = ((A+B)\*C)'임을 알 수 있다. (A'+B')\*C' 와 ((A\*B)+C)'는 A,B,C 모두 0일 때, B만 1일 때, A만 1일 때 1의 결과를 갖고 그 외에는 0의 값을 갖는다. (A'\*B')+C' 와 ((A+B)\*C)'는 B와 C가 1일 때, A와 C가 1일 때, 모두 1일 때 0의 값을 갖고, 그 외에는 1의 값을 갖는다. 이를 통해 두 식들은 서로 같은 결과를 갖는 것을 알 수 있다. 또한, 드모르간의 법칙이 3개의 input을 받았을 때에도 잘 적용이 가능한 것을 알 수 있고, 드모르간의 법칙을 통해 유도할 수 있음을 알 수 있다.

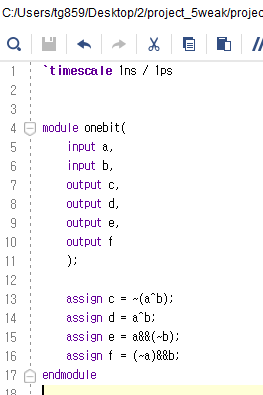
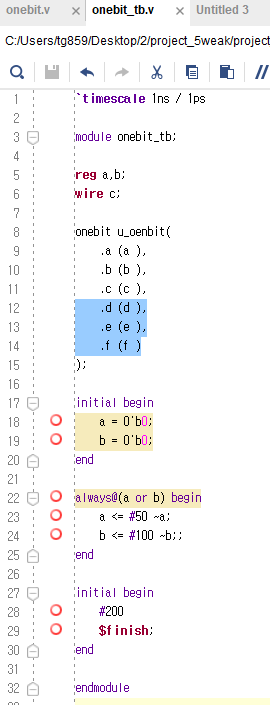
**4. 1Bit 비교기의 simulation 결과 및 과정에 대해서 설명하시오.**

**(2 input, 4 output)[진리표 작성]**

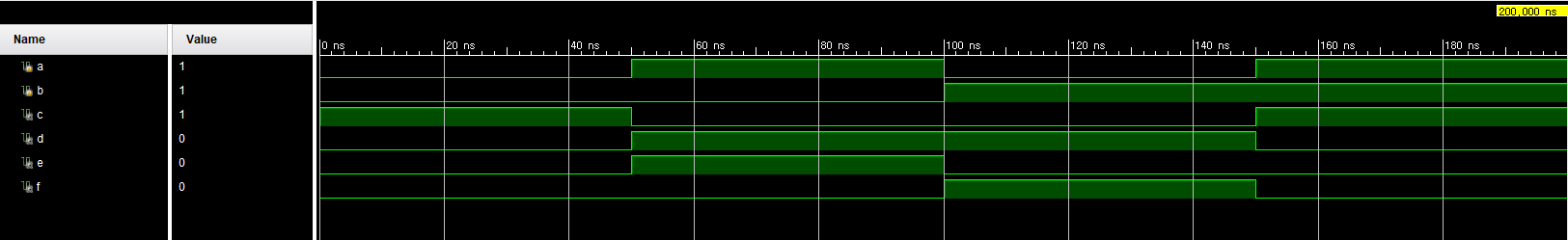
1. **Schematic**

****

1. **Design Source와 Testbench file**

**** ****

1. **Simulation 결과**

****

1. **진리표**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| A(in) | B(in) | A=B | A!=B | A>B | A<B |
| 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 |

1. **결과 및 구현 과정**

1 bit 비교기를 구현하기 위해 2개의 input a,b와 각각의 결과가 나오는 c,d,e,f를 정의하였다. c는 A=B로 XNOR연산을 통해 구현하였고, d는 A!=B로 XOR 연산을 통해 구현하였다. e는 A>B로 a AND !b를 통해 구현하였고, f는 A<B로 !a AND b를 통해 구현하였다. 해당 simulation 결과 4-3)의 이미지처럼 나왔고, 이를 바탕으로 진리표를 작성해본 결과 제대로 된 결과가 나왔음을 알 수 있다.

**5. 결과 검토 및 논의사항**

이번 실험을 통해 드모르간의 제 1법칙과 제 2법칙이 모두 유효함을 확인할 수 있었다. 이를 통해 논리 회로 설계 시 복잡한 논리식을 간소화하거나, 주어진 식을 다른 형태로 변환할 때 유용하게 사용할 수 있다. 또한 효율적인 하드웨어 구현을 가능하게 하며, 설계의 복잡성과 제조 비용을 줄일 수 있을 것이다.

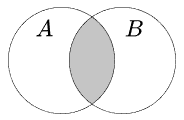
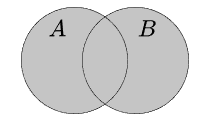
3가지 input을 받은 논리식도 (A'+B')\*C' = ((A\*B)+C)'와 (A'\*B')+C' = ((A+B)\*C)' 임을 실험을 통해 알 수 있었다. 이를 통해 복잡한 논리식도 드모르간의 법칙을 통해 간소화 될 수 있음을 알 수 있다. 또한 설계의 유연성을 증가시키고 최적화된 논리 회로를 설계할 수 있을 것이다.

1 bit 비교기의 구현을 통해 대소 비교를 논리 연산자를 이용해 나타낼 수 있음을 알 수 있었다. 해당 결과를 통해 여러 응용 문제도 해결할 수 있을 것이다.

**6. 추가 이론 조사 및 작성.**

우리는 지금까지 Verilog를 이용한 simulation과 FPGA를 이용한 실습을 통해 드모르간의 법칙이 성립함을 증명하였다. 이 외에도 여러 증명 방법이 존재한다. 대표적인 방법은 벤 다이어그램으로 그려서 증명하는 방식과 가정을 통한 증명이 있다.

1. 벤 다이어그램을 통한 증명



(A+B)’ = A’\*B’ (A\*B) = A’+B’

1. 가정을 통한 증명

각각의 케이스에서 반대의 경우는 비슷한 방식으로 증명 가능하므로 양뱡향 증명이 완료된다.