5주차 결과보고서

전공: 수학과 학년: 4학년 학번: 20171273 이름: 심현우

1. **실험목적**

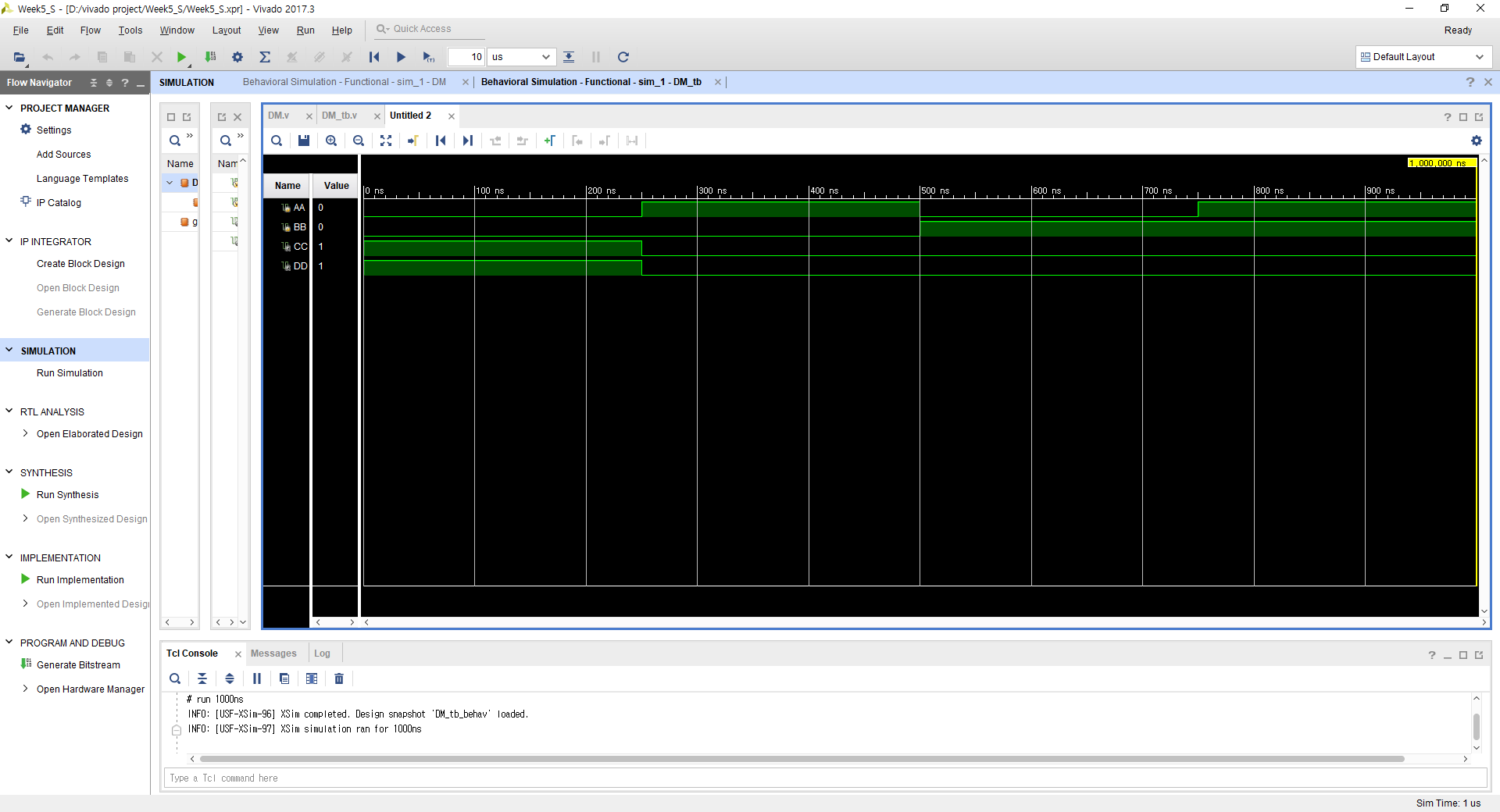
회로의 간소화를 위한 De-Morgan의 법칙을 적용하고 결과를 비교하는 실험을 진행했다. 적당한 equation에 대한 NOT연산을 계산해보고 Verilog 코딩을 통해 실제로 결과값이 똑같이 나오는지 비교하는 과정을 수행했다. 또한 1bit에 대한 비교기, 즉 0과 1의 input에 대해 서로를 비교하여 같거나 다르거나 어떤 것이 더 큰지를 비교해 맞으면 1, 틀리면 0을 출력하는 장치도 코드를 통해 구현했다.

1. **De-Morgan의 제 1, 2 법칙의 simulation 결과 및 과정에 대해서 설명하시오.**
2. **1법칙**

텍스트, 영수증, 스크린샷이(가) 표시된 사진

자동 생성된 설명테이블이(가) 표시된 사진

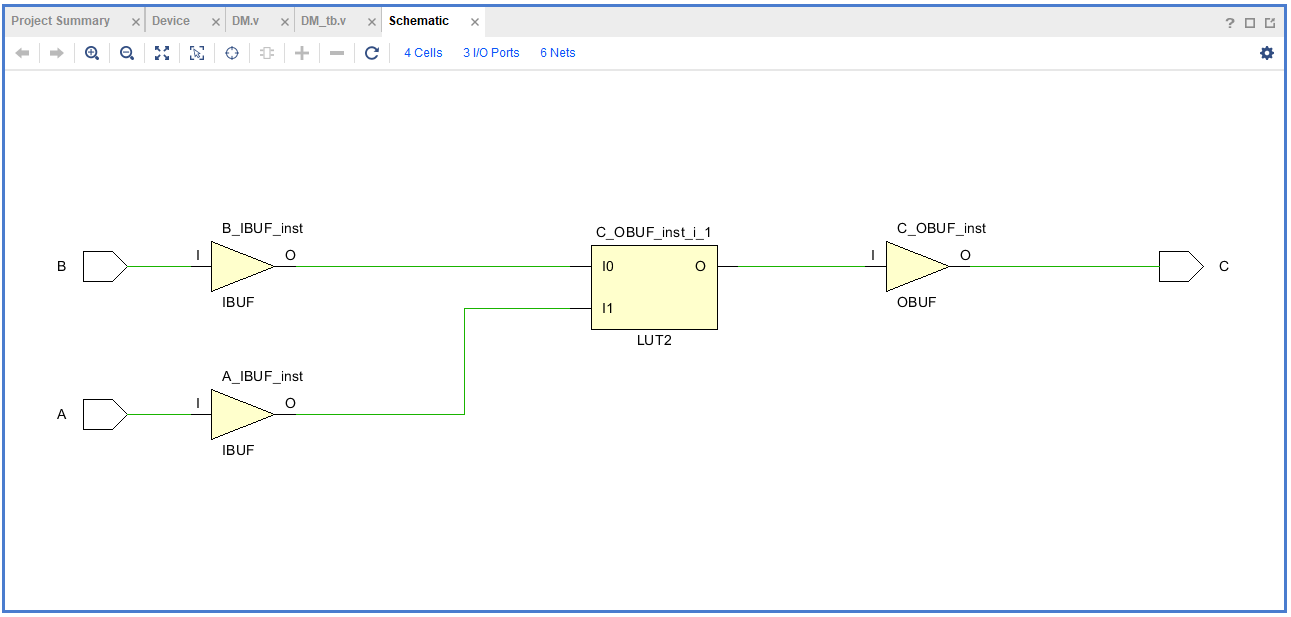
자동 생성된 설명

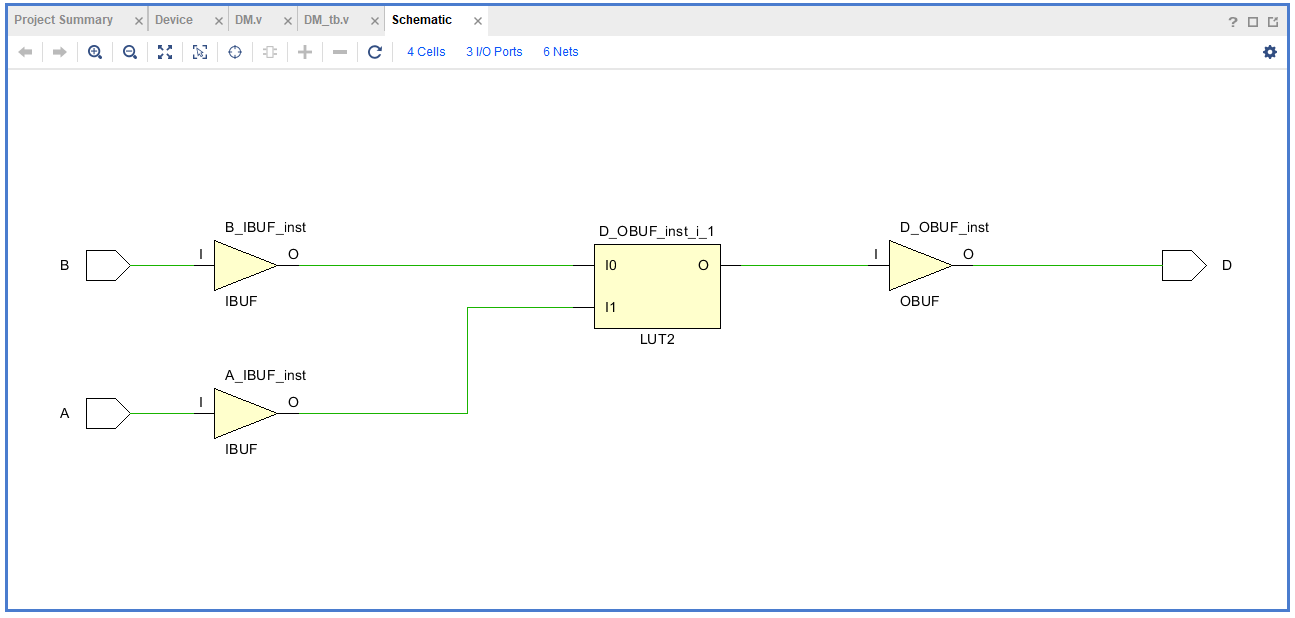


위의 코드부터 살펴보자. De-Morgan 제 1법칙을 확인하기 위한 게이트을 구성했다. 위의 A, B가 input으로, C, D는 각각 법칙을 적용하지 않은 equation과 적용한 equation에 대한 output이다. Assign을 보면 C=~(A|B)로 Boolean 식 전체에 NOT 연산이 붙어있고, D=~A&~B로 De-Morgan법칙을 이용해 Boolean 식을 정리한 것이다. 이를 simulation으로 확인해보면 같은 input A, B에 대해 같은 결과를 출력하는 것을 C와 D를 통해 알 수 있다. 이를 schematic을 통해 확인해보자.



(A)와 (B)는 구성하고자 했던 schematic이다. vivado에서 작성한 코드를 통해 schematic의 결과를 확인해보자.





Vivado schematic에서도 두 Boolean 식이 같음을 확인할 수 있다. 또한 제 1법칙에서 다룬 Boolean 식은 4주차에 다뤘던 NOR게이트와 동일하다. 이는 boolean식이나 schematic으로 확인할 수 있다. Simulation의 결과를 진리표를 작성하여 한번 더 확인해보자.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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 |

1. **2법칙**

텍스트, 영수증, 스크린샷이(가) 표시된 사진

자동 생성된 설명 텍스트, 영수증이(가) 표시된 사진

자동 생성된 설명

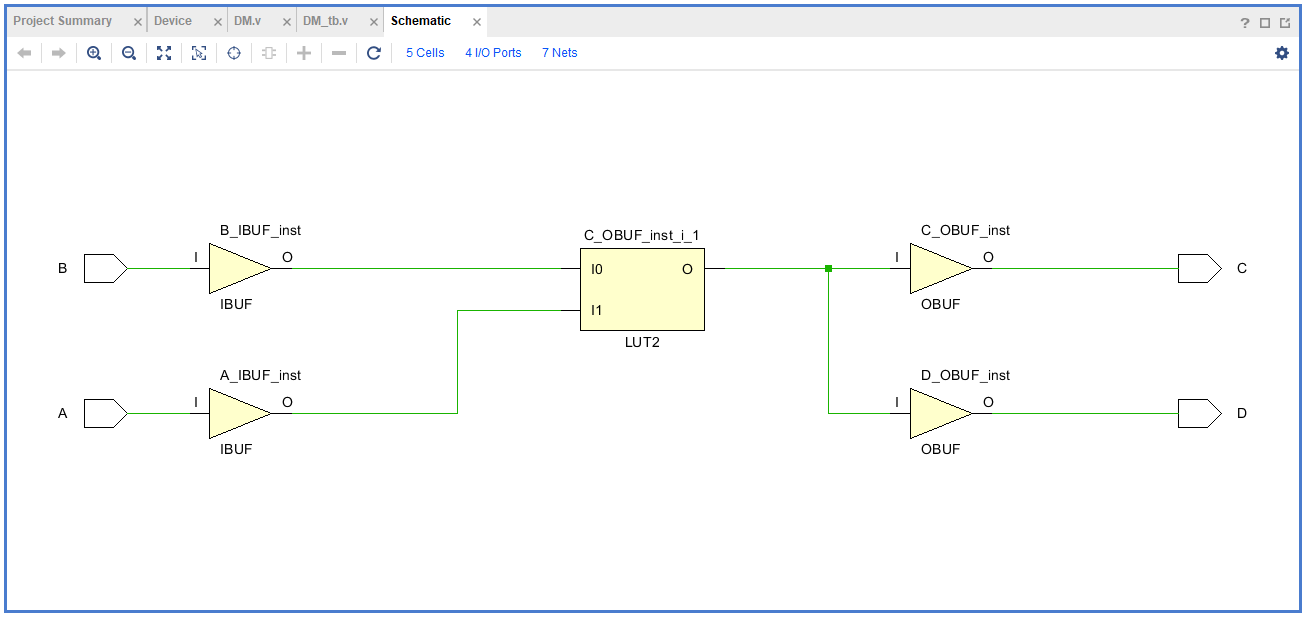
텍스트, 스크린샷, 전자기기, 컴퓨터이(가) 표시된 사진

자동 생성된 설명

위의 코드부터 살펴보자. De-Morgan 제 2법칙을 확인하기 위한 게이트을 구성했다. 위의 A, B가 input으로, C, D는 각각 법칙을 적용하지 않은 equation과 적용한 equation에 대한 output이다. Assign을 보면 C=~(A&B)로 Boolean 식 전체에 NOT 연산이 붙어있고, D=~A|~B로 De-Morgan법칙을 이용해 Boolean 식을 정리한 것이다. 이를 simulation으로 확인해보면 같은 input A, B에 대해 같은 결과를 출력하는 것을 C와 D를 통해 알 수 있다. 이를 schematic을 통해 확인해보자.



(A)와 (B)는 구성하고자 했던 schematic이다. vivado에서 작성한 코드를 통해 schematic의 결과를 확인해보자.



Vivado schematic에서도 두 Boolean 식이 같음을 확인할 수 있다. 또한 제 2법칙에서 다룬 Boolean 식은 4주차에 다뤘던 NAND게이트와 동일하다. 이는 boolean식이나 schematic으로 확인할 수 있다. Simulation 결과를 진리표를 통해 한번 더 확인해보자.

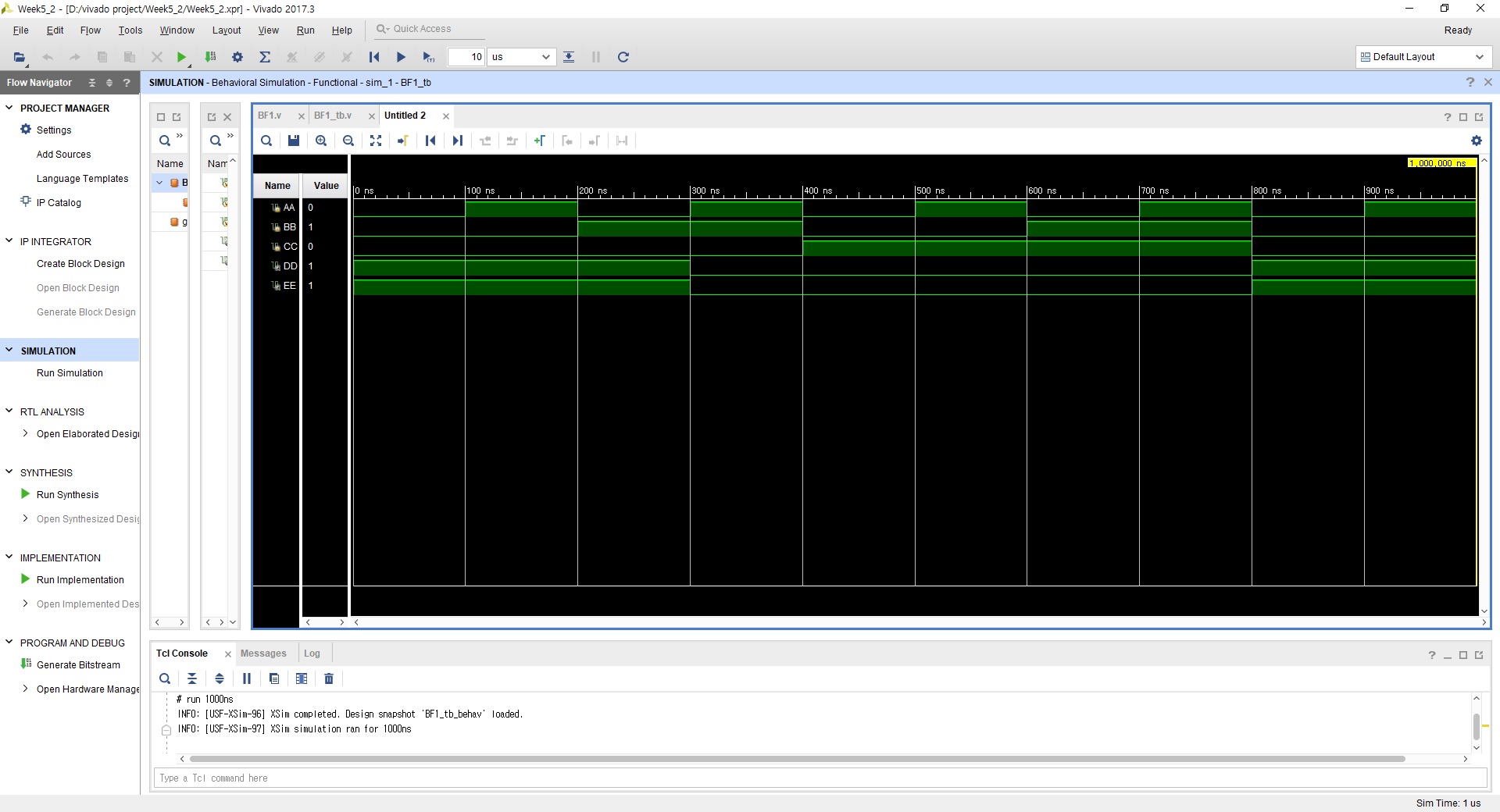
|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A | B | A|B | A&B | ~A | ~B | ~(A|B) | ~A&~B | ~(A&B) | ~A|~B |
| 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

1. **(A’+B’)\*C’=((A\*B)+C)’의 simulation 결과 및 과정에 대해서 설명하시오.**
2. **(A’+B’)\*C’=((A\*B)+C)’**

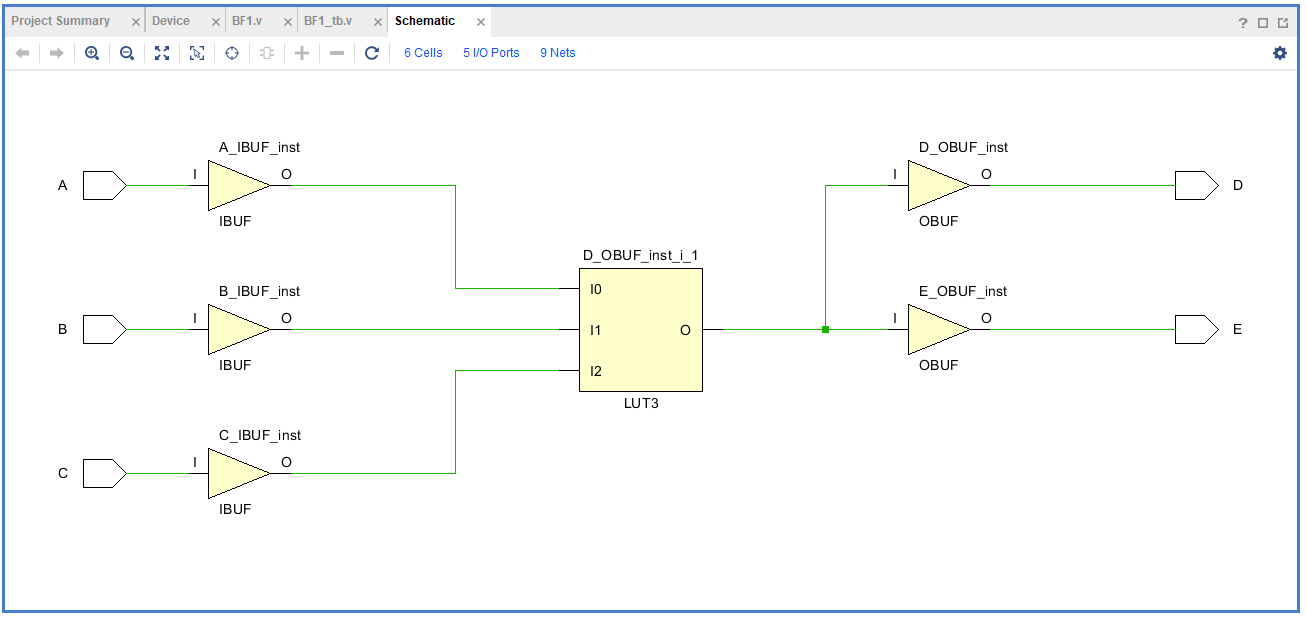
텍스트이(가) 표시된 사진

자동 생성된 설명 텍스트, 영수증, 스크린샷이(가) 표시된 사진

자동 생성된 설명



위의 코드부터 살펴보자. 주어진 Boolean 식이 동일하다는 것을 확인하기 위한 게이트을 구성했다. 위의 A, B, C 가 input으로, D, E는 각각 좌변과 우변에 대한 output이다. Assign을 보면 D=(~A|~B)&~C로 검증하고자 하는 식의 좌변에 해당하고, E=~( (A&B)|C )로 우변의 식의 결과를 나타낸 것이다. 이를 simulation으로 확인해보면 같은 input A, B에 대해 같은 결과를 출력하는 것을 D와 E를 통해 알 수 있다. 이를 schematic을 통해 확인해보자. 우변의 식을 De-Morgan의 정리를 이용하여 간소화하게 되면 Boolean 식 자체로도 두 식이 동일한 것을 알 수 있다. 이를 vivado schematic을 통해 한번 더 확인해 보자.



schematic으로 보면 D와 E의 결과가 같은 것도 확인할 수 있다. 마지막으로 simulation 결과를 진리표로 작성하여 비교해보자.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| In A | In B | In C | Out D | Out E |
| 0 | 0 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 |

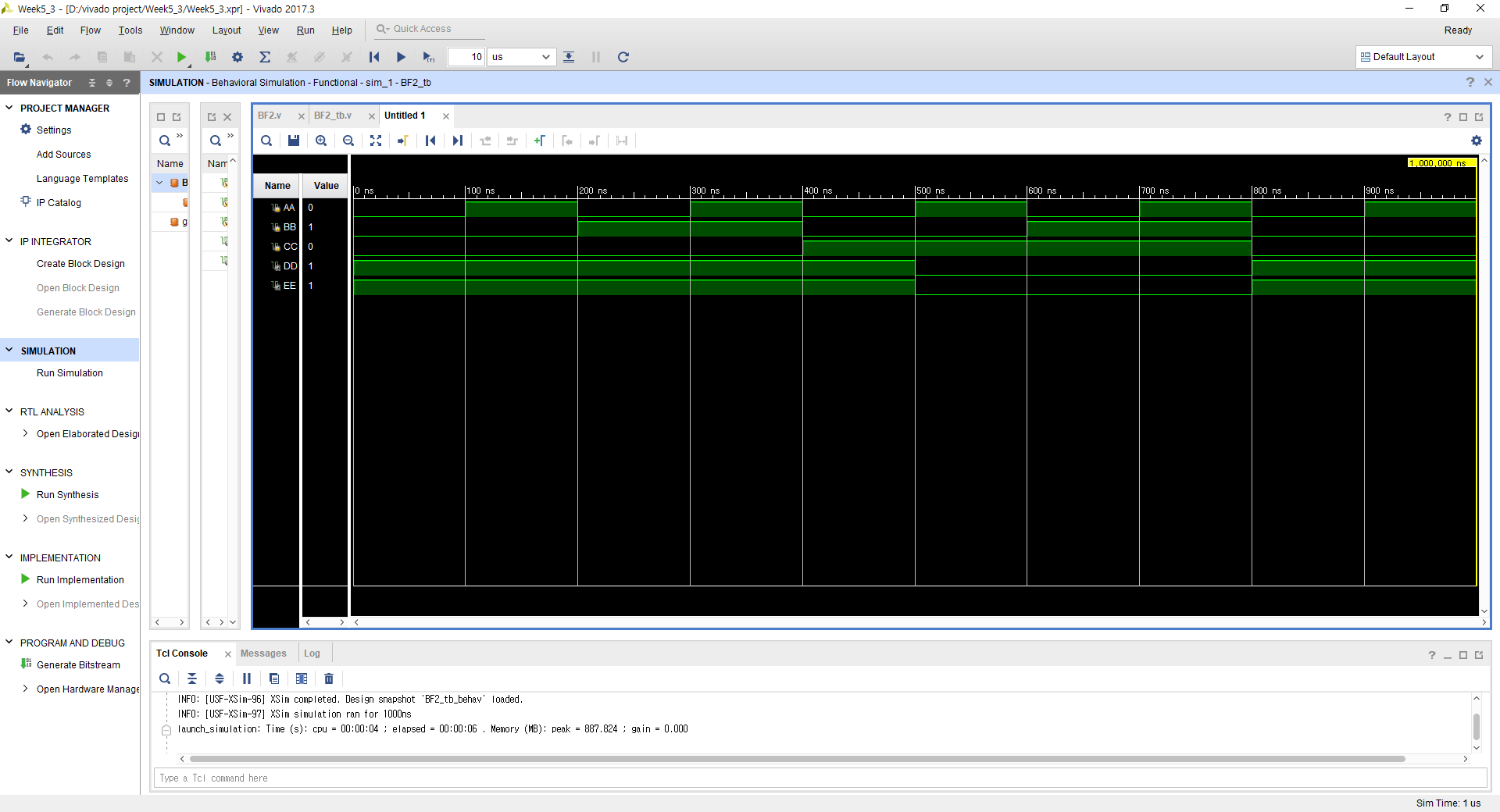
두 boolean식의 결과인 D와 E는 동일한 결과를 출력하는 것을 확인할 수 있다.

1. **(A’\*B’)+C’=((A+B)\*C)’**

텍스트이(가) 표시된 사진

자동 생성된 설명텍스트, 영수증, 스크린샷이(가) 표시된 사진

자동 생성된 설명

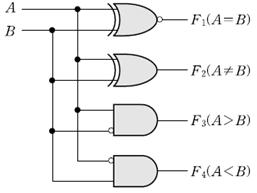


위의 코드부터 살펴보자. 주어진 Boolean 식이 동일하다는 것을 확인하기 위한 게이트을 구성했다. 위의 A, B, C 가 input으로, D, E는 각각 좌변과 우변에 대한 output이다. Assign을 보면 D=(~A&~B)|~C로 검증하고자 하는 식의 좌변에 해당하고, E=~( (A|B)&C )로 우변의 식의 결과를 나타낸 것이다. 이를 simulation으로 확인해보면 같은 input A, B에 대해 같은 결과를 출력하는 것을 D와 E를 통해 알 수 있다. 이를 schematic을 통해 확인해보자. 우변의 식을 De-Morgan의 정리를 이용하여 간소화하게 되면 Boolean 식 자체로도 두 식이 동일한 것을 알 수 있다. 이를 진리표를 통해 한번 더 확인해 보자.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| In A | In B | In C | Out D | Out E |
| 0 | 0 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 |
| 0 | 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 |

두 boolean식의 결과인 D와 E는 동일한 결과를 출력하는 것을 확인할 수 있다.

1. **1bit 비교기의 simulation 결과 및 과정에 대해서 설명하시오.**

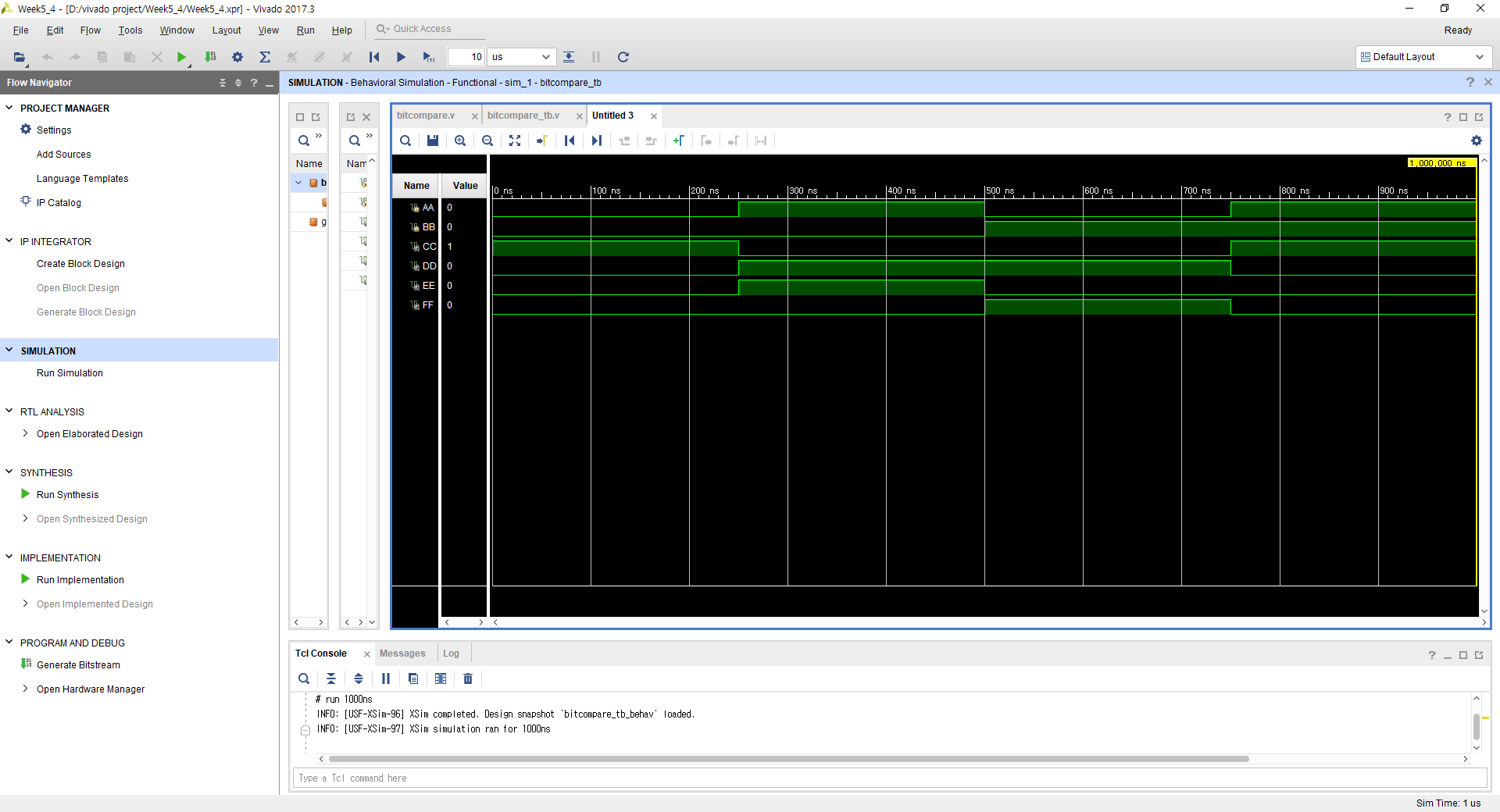


1bit 비교기의 논리회로는 위와 같다. F1먼저 보면 A=B일 때만 1의 값을 출력한다. 이는 서로 다른 값이 input으로 들어왔을 때 1의 값을 출력하는 XOR게이트의 마지막에 NOT 연산을 수행하여 값이 같을 때 1의 값을 출력하도록 한다. 두번째 F2는 F1과 반대 상황으로 XOR게이트와 같다. F3는 input A가 input B보다 클 때 1의 값을 출력하는 게이트로 A와 ~B를 AND게이트를 사용하여 구성하였다. F4는 F3와 반대의 상황으로 ~A와 B를 AND게이트를 사용하여 구성하였고 B가 A보다 클 때 1의 값을 출력한다.

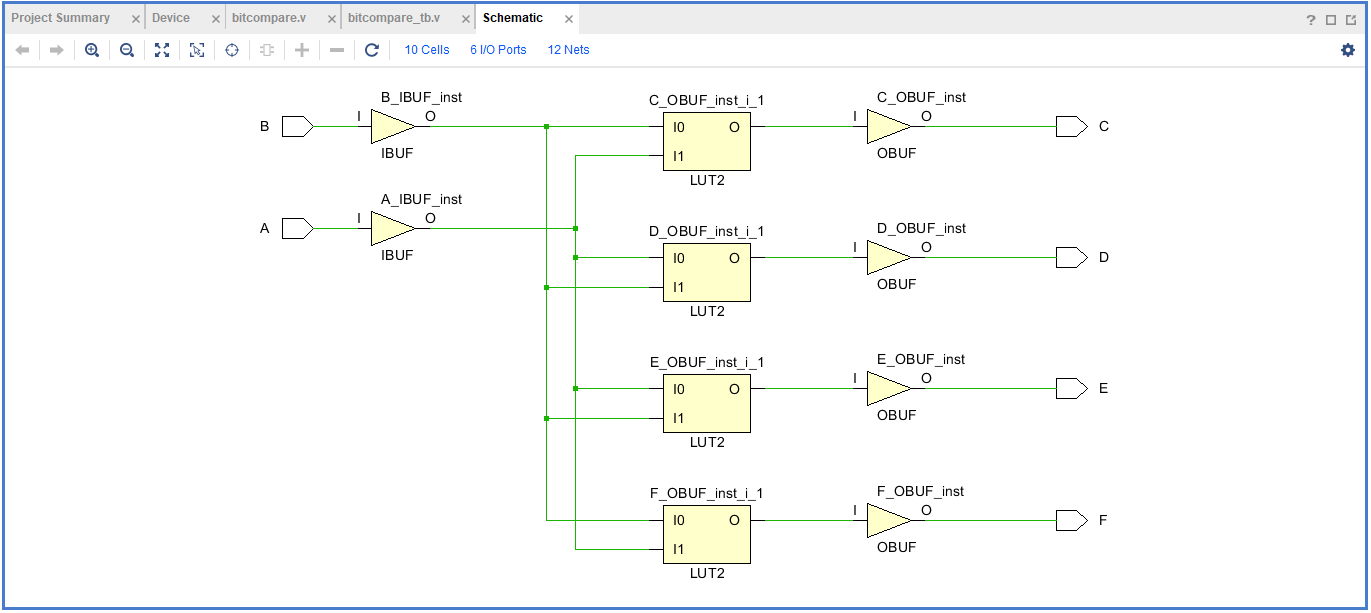
텍스트이(가) 표시된 사진

자동 생성된 설명텍스트, 영수증, 스크린샷이(가) 표시된 사진

자동 생성된 설명



위의 코드부터 살펴보자. 위의 사진에 따라 구성한 논리회로를 코드로 작성했다. 위의 A, B가 input으로 비교하는 대상이고, C, D, E, F는 각각 F1, F2, F3, F4에 대한 output이다. Assign을 보면 C=~ ( (~A&B) | (A&~B) )로 XOR게이트에 NOT을 붙여 구성했다. D= (~A&B) | (A&~B)로 F2 XOR게이트를 만들었다. E = A&~B로 F3, F = ~A&B로 F4를 만들었다. 이를 simulation으로 확인해보면 input A, B의 bit에 대해 같거나 다르거나, 대소를 비교하는 게이트가 잘 구성된 것을 알 수 있다. 이를 schematic을 통해 확인해보자.



처음 첨부한 논리회로 사진처럼 게이트가 잘 구성되어 있는 것을 확인할 수 있다. 진리표로 1bit 비교기에 대한 결과를 작성해 봤다.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| In A | In B | Out C(A=B) | Out D(A!=B) | Out E(A>B) | Out F(A<B) |
| 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 | 0 | 0 |

1. **결과 검토 및 논의사항**

이번 실험에서는 De-Morgan을 통해 Boolean 식의 간소화를 적용하고 실제로 결과값을 비교하는 실험을 진행했다. De-Morgan의 제 1, 2 법칙부터 복잡한 boolean식을 법칙을 적용했을 때와, 하지 않았을 때를 비교했다. 직접 계산하여 결과를 비교하였을 때와 verilog 코드를 작성하여 simulation으로 결과를 비교하였을 때 동일하게 결과가 나왔다. 따라서 식을 간소화함에 있어 De-Morgan 법칙을 유용하게 사용할 수 있다. 또한 1bit 비교기를 만들어 input 값에 대한 비교하는 작업도 진행했다. 이는 지금까지 진행했던 AND, NOT, XOR 등 기본 게이트를 이용하여 구현할 수 있었다. 이 비교기를 통하여 후에 input의 값을 비교해야 하는 상황에 유용하게 사용할 수 있을 것이다.

1. **추가 이론 조사 및 작성**











(참조 : introduction to logic and computer design)

De-Morgan의 법칙 이외에도 Boolean 식의 간소화를 위한 방법이다. 위 식들은 모든 input값에 대한 output을 진리표로 작성하여 양 변이 동일한 결과를 도출하는 것을 확인할 수 있다. De-Morgan처럼 자주 쓰이는 것이 위의 8번과 9번의 distributive와 adjacency이다. Distributive의 경우 우리가 흔히 알고 있는 사칙연산의 분배법칙과 비슷하다. Adjacency는 distributive을 이용하여 정리하면 따라오는 정리이다.