컴퓨터공학실험2 5주차 결과 보고서

전공: 컴퓨터공학 학년: 2학년 학번: 20201635 이름: 전찬

**0. 목차**

1. 실험 목적

2. De-Morgan의 제 1, 2 법칙의 simulation 결과 및 과정 설명

3. (A’+B’)C’ = (AB+C)’의 simulation 결과 및 과정 설명(+와 \* 위치를 바꾼 형태도 포함)

4. 1bit 비교기의 simulation 결과 및 과정 설명

5. 결과 검토 및 논의 사항

**1. 실험 목적**

De Morgan의 제 1, 2 법칙에 대해 이해한다. 이를 토대로, 어떤 논리식에 NOT을 적용할 때의 결과를 파악해본다. 1bit 비교기를 직접 구현해보며 실제 비교 logic에서 ==, !=, >, < 가 어떻게 구현되는지 파악해본다.

**2. De Morgan의 제 1, 2 법칙의 simulation 결과 및 과정**

회로 이론에서 De Morgan 법칙은 아래와 같다.

(1) ~(XY) = (~X)+(~Y)

(2) ~(X+Y) = (~X)(~Y)

이를 직접 구현한 코딩과 simulation을 진행한 결과는 아래와 같다.(A, B는 input이며 sim1\_1, sim1\_2, sim1\_3, sim1\_4는 각각 제 1법칙의 좌변과 우변, 제 2법칙의 좌변과 우변이다.)

(1-1) assign sim1\_1 = ~(A&B);

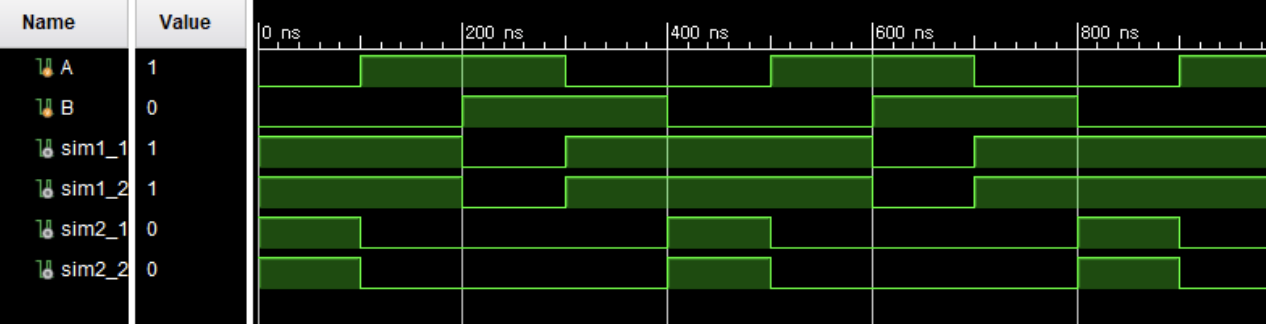
(1-2) assign sim1\_2 = (~A)|(~B);

(2-1) assign sim2\_1 = ~(A|B);

(2-2) assign sim2\_2 = (~A)&(~B);

<식을 표현한 코드>

또한 simulation 결과는 아래와 같다.



<De-Morgan 법칙의 simulation 결과>

결과적으로 위 De-Morgan 법칙이 성립함을 알 수 있으며, 이 법칙을 논리식 간소화 등 다양한 분야에서 적용할 수 있음 또한 파악할 수 있다.

**3. De-Morgan 법칙을 활용한 식의 simulation 결과 및 과정 설명**

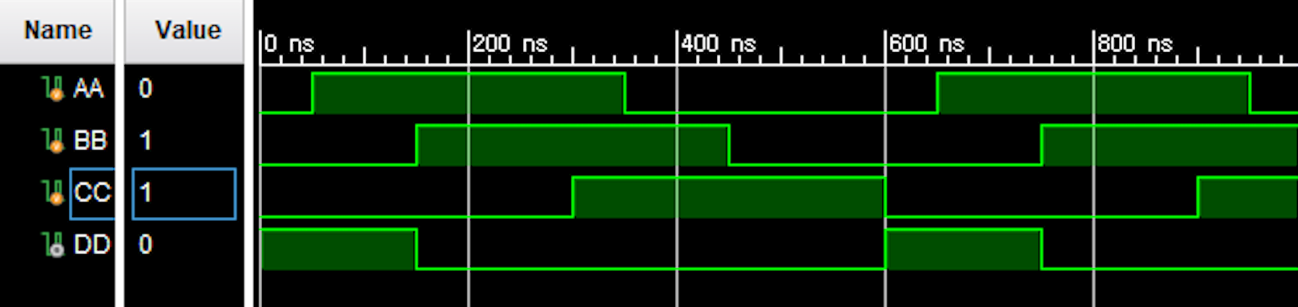
이번 실습에서는 총 4가지의 De-Morgan 법칙에 관련된 식을 직접 구현해서 simulation 해보는 실습을 진행했었다. 그 중에서 구현한 첫 번째 식은 (A’+B’)C’ = (AB+C)’ 이며, 좌변과 우변이 동일하다는 것을 De-Morgan 법칙을 통해서 파악할 수 있다. 이는 아래와 같다.

우선 우변을 기준으로 De-Morgan 법칙을 적용하면, (AB+C)’ = (AB)’C’ 이다.

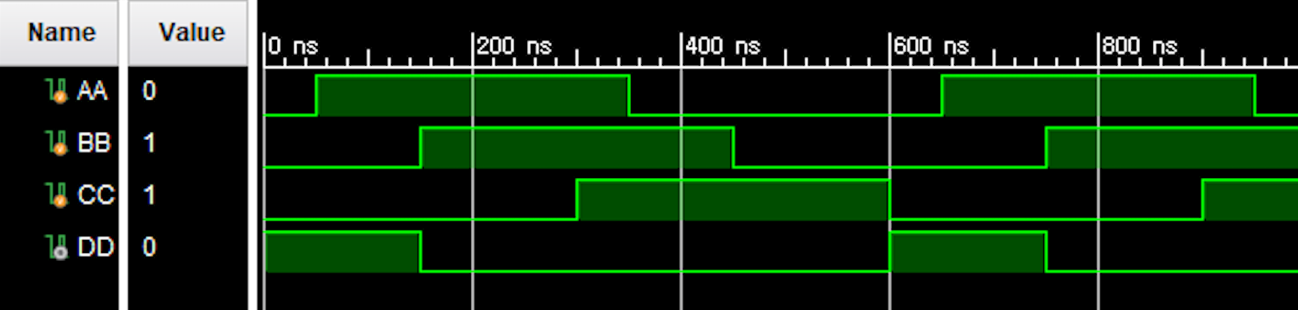
(AB)’에 다시 De-Morgan 법칙을 다시 적용하면, (AB)’ = A’+B’ 이다.

마지막으로 위 두 결과를 결합하면, (AB+C)’ = (A’+B’)C’임을 확인할 수 있다.

위 식에서의 simulation 결과는 아래와 같다.



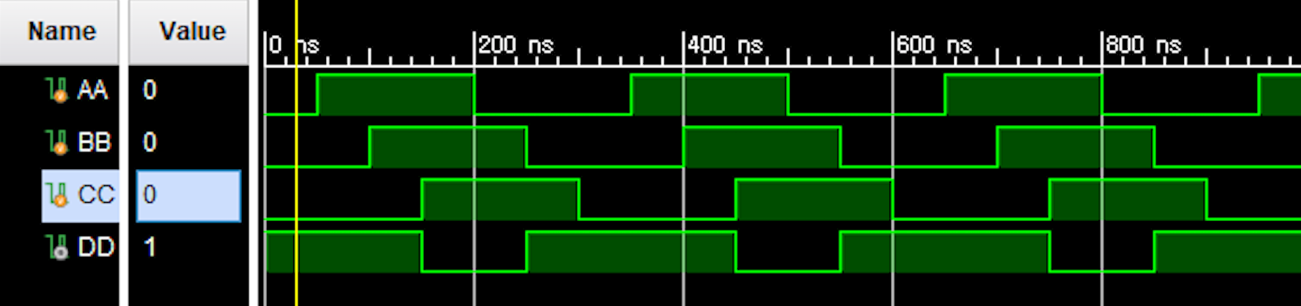
<첫 번째 식의 좌변(1-A)의 simulation 결과>



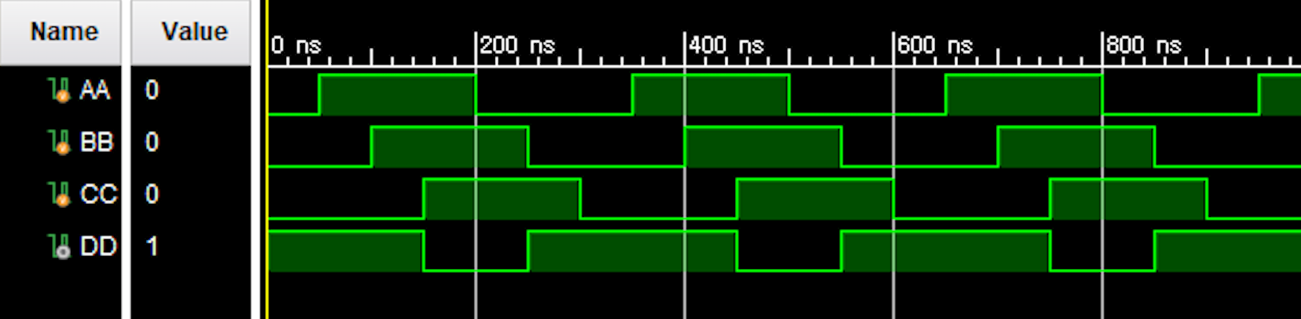
<첫 번째 식의 우변(1-B)의 simulation 결과>

두 번째 식은 A’B’+C’ = ((A+B)C)’ 이다. 이 또한 De-Morgan 법칙을 통해서 동일함을 파악할 수 있는데, ((A+B)C)’ = (A+B)’+C’ = A’B’+C’ 형태로 좌변과 우변이 동일함을 알 수 있다.

두 번째 식의 simulation 결과는 아래와 같다.



<두 번째 식의 좌변(2-A)의 simulation 결과>



<두 번째 식의 우변(2-B)의 simulation 결과>

추가적으로 각 simulation에서 module의 assign 식은 아래와 같다.

(1-A) assign out = ((~in1) | (~in2)) & (~in3);

(1-B) assign out = ~((in1 & in2) | in3);

(2-A) assign out = ((~in1) & (~in2)) | (~in3);

(2-B) assign out = ~((in1 | in2) & in3);

<식을 표현한 코드>

**4. 1bit 비교기의 simulation 결과 및 과정 설명**

1bit 비교기(comparator)은 두 1bit의 input(input A, B)을 받았을 때, 4가지 경우(A=B, A!=B, A>B, A<B)에 대해서 참/거짓을 판단해주는 회로이다. 각 경우의 구현을 보자면, A=B인 경우는 A = 1, B = 1, A = 0, B = 0 일 때 output이 1이 되도록 만들어주면 된다. 이는 XOR gate에 다시 not을 적용한 형태로 쉽게 구현할 수 있다. A!=B일 때는 A=B인 경우에서 다시 NOT을 적용하면 된다. 따라서 XOR gate로 판단할 수 있다. A>B인 경우는 A = 1, B = 0일 때만 output이 1을 갖도록 회로를 구현하면 된다. 이는 AND gate와 NOT gate를 활용하여, A AND (NOT B) 형태로 구현하면 된다. 또한 반대 경우에는, NOT을 A에 붙여서 바꾸면 된다.

이에 대한 Verilog 코딩은 아래와 같다.

(1) assign out1 = ~(A^B); // A=B

(2) assign out2 = A^B; // A!=B

(3) assign out3 = A&(~B); // A>B

(4) assign out4 = (~A)&B; //A<B

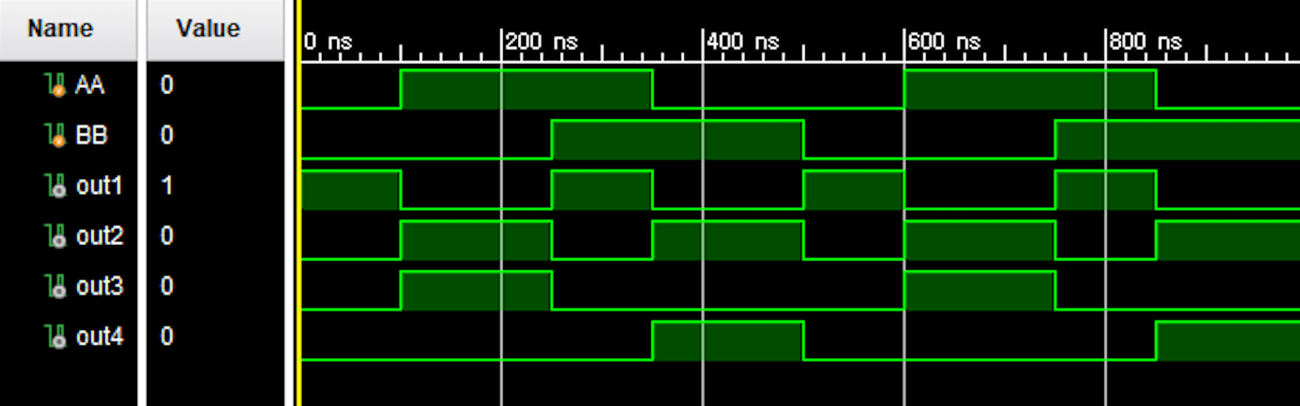
<1bit 비교기의 Verilog 코딩>

1bit 비교기의 진리표는 아래와 같다.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1bit 비교기 | | | | | |
| input | | output | | | |
| A | B | out1 | out2 | out3 | out4 |
| 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 0 |

<1bit 비교기의 진리표>

또한 이에 대한 simulation 결과는 아래와 같다.



<1bit 비교기의 simulation 결과>

**5. 결과 검토 및 논의 사항**

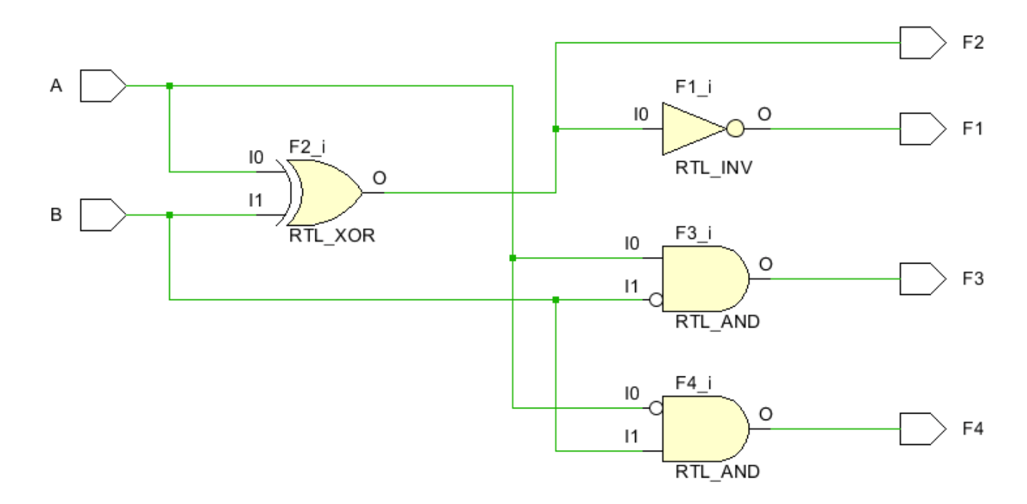
이번 실습을 통해서, De-Morgan 법칙이 다양한 회로에서도 공통적으로 성립됨을 파악할 수 있으며, 1bit comparator을 제작해서 두 input이 존재할 때, 실제로 어떤 gate를 사용해서 회로를 비교할 수 있는지를 파악해보기도 했다. 이때 실습에서 제작한 1bit 비교기에서는 총 7개의 gate를 사용했다고 할 수 있는데, 이를 논리식 간소화(gate의 수를 줄이기)할 수 있는지 생각을 하게 되었다. 따라서 다음과 같은 형태를 생각했다.

1. 우선 A>B, A<B 형태를 구현한다.

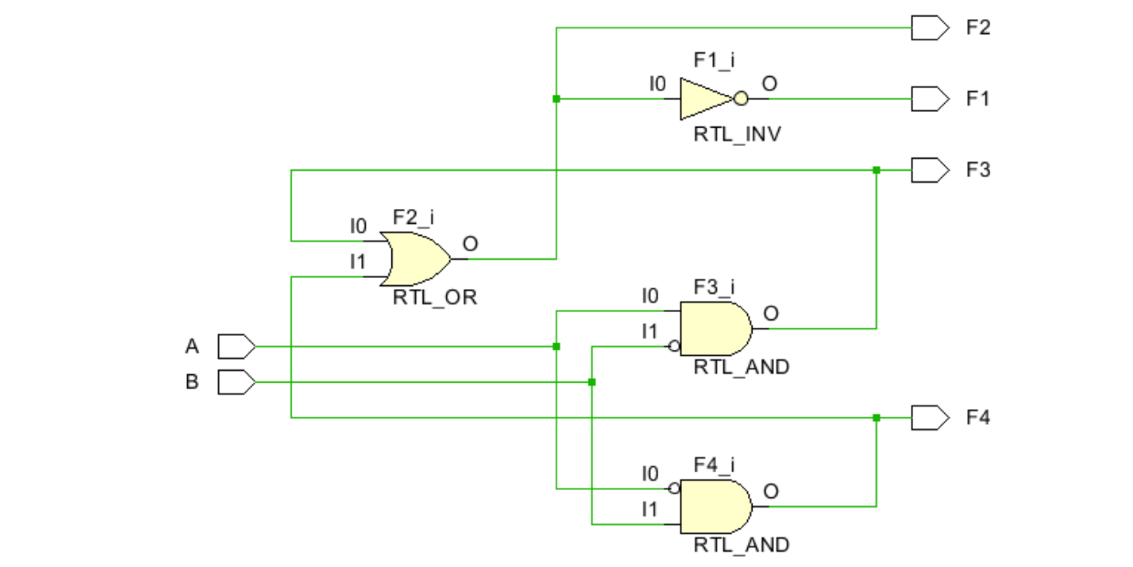
2. 위 두 형태를 OR gate로 묶어 A!=B를 만들어낸다.

3. A!=B에 NOT gate를 이용해 A=B를 만들어낸다.

위와 같이 하면, 1.에서 2가지 gate, 2.에서 1가지 gate, 3.에서 1가지 gate를 사용해서 총 6가지 gate로 동일한 비교기를 제작할 수 있을 것이다. 이에 대해서 verilog에서의 schematic은 아래와 같다.



<첫 번째 구현에서의 schematic>



<두 번째 구현에서의 schematic>

위 schematic 결과를 봤을 때는 첫 번째 구현에서도 6가지의 gate를 사용해서 1bit 비교기를 만들었다는 것을 알 수 있다. 결과적으로는, Verilog 프로그램 자체에서 우리가 구현한 회로를 자동으로 최적화(최적화 까지는 아니지만, 더 효율적으로) 해준다는 것을 파악할 수 있었다.