5주차 결과보고서

전공: 컴퓨터공학과 학년: 2학년 학번: 20191619 이름: 이동석

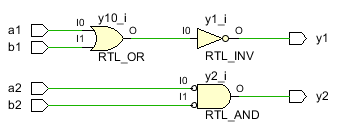
**1. 목적**

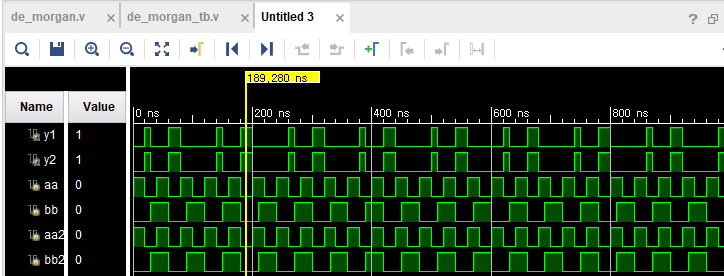
De-Morgan의 정리를 이해하고, NOR/NAND와 비교해 어떤 점이 다른지 이해한다. 또한, 드모르간 정리를 적용하기 전 식과 적용한 식의 시뮬레이션 출력 값이 같은지 직접 코드 구현을 통해 비교하고 동작해 본다. 1-bit 비교기에 대해 알아본다. 비교기의 진리표와 시뮬레이션 결과가 같은 지 확인하고 동작을 구현해본다.

**2. De-Morgan 제 1,2법칙**

제 1법칙

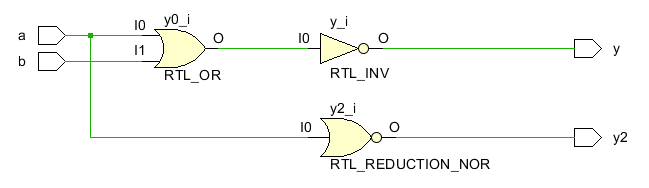
**텍스트이(가) 표시된 사진

자동 생성된 설명**

****

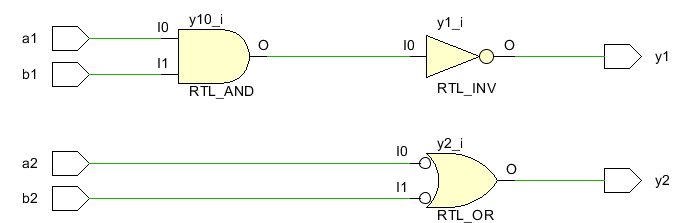
드로므르간의 제 1법칙이다. 알다시피 제 1법칙은 논리합 -> 논리곱으로 간다. 다시말해서, 표현식을 코드로 바꾸면 ~(a1 | b1 ) = (~a2) & (~b2) 이다. 이 두 회로의 시뮬레이션 결과를 돌려보면 역시 완벽하게 동일한 것을 알 수 있다. 입력 받는 논리 값에 1이 포함되어있다면, 항상 0을 출력하며 모두 0이라면 1을 출력한다.

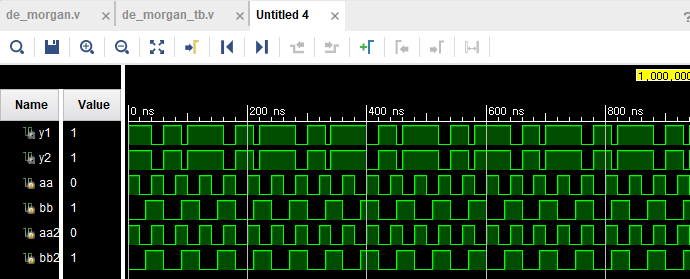
또한, 회로의 모습이 어디선과 본 듯한 모습일텐데 이는 NOR 게이트와 완벽하게 일치하기 때문이다. NOR게이트의 스케메틱은 아래와 같다



제 2법칙

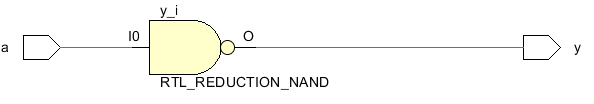
**텍스트이(가) 표시된 사진

자동 생성된 설명**

****

드로므르간의 제 2법칙 이다. 제 2법칙은 논리곱 -> 논리합으로 간다. 코드를 작성하면, ~(a1 & b1 ) = (~a2) | (~b2)이다. 두 회로의 시뮬레이션을 돌려보면 결과 역시 완벽하게 동일한 것을 알 수 있다. 모든 입력 값이 1이라면, 항상 0을 출력한다.

또한, 제 1법칙과 비슷하게 회로의 모습이 어디선과 본 듯한 모습이다. 이는 NAND 게이트와 완벽하게 일치한다. NAND게이트의 스케메틱은 아래와 같다.



드모르간 진리표

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **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 |

**NOR NAND**

테이블이(가) 표시된 사진

자동 생성된 설명테이블이(가) 표시된 사진

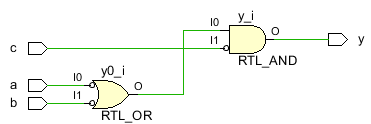
자동 생성된 설명

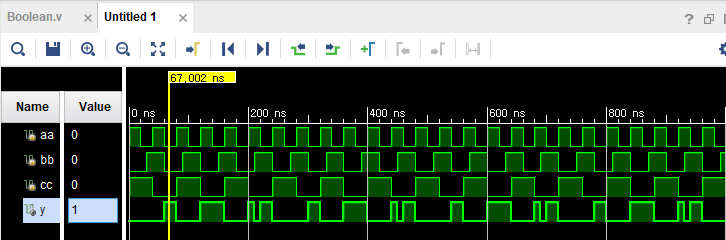
NOR게이트와 NAND게이트의 진리표는 위와 같다. 이를 앞서 작성한 드모르간 법칙의 진리표와 비교해 보면, NOR는 (A+B)’ 와 완벽하게 동일함을 알 수 있고, NAND의 경우 (A\*B)’와 동일함을 알 수 있다. 이는 다시 드모르간 법칙에 의해 각각 A’\*B’와 A’+B’와 동일하다.

**3. Boolean Function**

(A’+B’)\*C’

**텍스트이(가) 표시된 사진

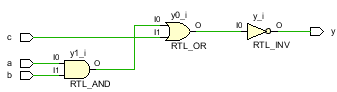
자동 생성된 설명**

****

|  |  |  |  |
| --- | --- | --- | --- |
| **IN A** | **IN B** | **IN C** | **OUT D** |
| 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 |

((A\*B)+C)’

**테이블이(가) 표시된 사진

자동 생성된 설명**텍스트이(가) 표시된 사진

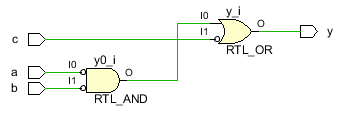
자동 생성된 설명

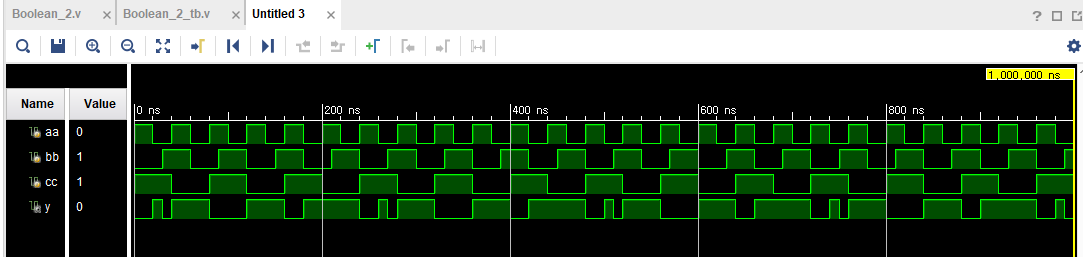
|  |  |  |  |
| --- | --- | --- | --- |
| **IN A** | **IN B** | **IN C** | **OUT D** |
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 |

위의 두코드는 사뭇 다르지만, 시뮬레이션 결과는 완벽하게 일치한다. 또한, 마찬가지로 진리표 역시 동일하다. (테스트 벤치 파일에서 각 변수들은 같은 시간에 변한다.) 두 식의 동일함은 앞서 배운 드모르간 법칙에 의해서도 보일 수 있다. 진리표를 보면 A와 B가 둘 중 하나 이상이 0이고 C도 0일 때 출력 값 D가 1이 됨을 알 수 있다.

(A’\*B’)+C’

테이블이(가) 표시된 사진

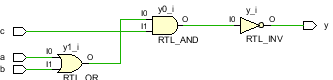
자동 생성된 설명



|  |  |  |  |
| --- | --- | --- | --- |
| **IN A** | **IN B** | **IN C** | **OUT D** |
| 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 |

((A+B)\*C)’

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트, 실내, 스크린샷이(가) 표시된 사진

자동 생성된 설명

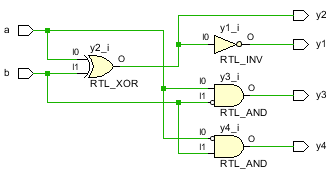
|  |  |  |  |
| --- | --- | --- | --- |
| **IN A** | **IN B** | **IN C** | **OUT D** |
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 |

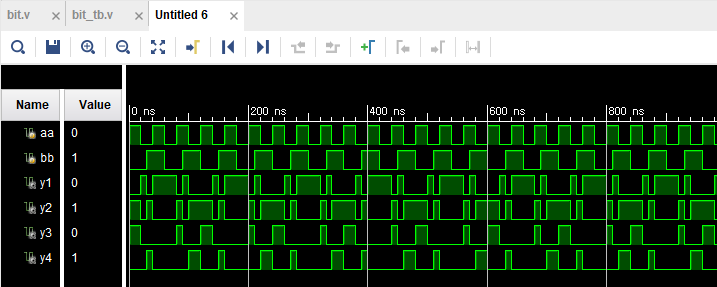
마찬가지로 위의 두코드는 사뭇 다르지만, 시뮬레이션 결과는 완벽하게 일치한다. 또한, 마찬가지로 진리표 역시 동일하다. (테스트 벤치 파일에서 각 변수들은 같은 시간에 변한다.) 두 식의 동일함은 앞서 배운 드모르간 법칙에 의해서 보일 수 있다. 진리표를 보면 A와 B가 둘 모두가 0이거나 C가 0일 때 출력 값 D가 1이 됨을 알 수 있다.

결론적으로 (A’+B’)\*C’ = ((A\*B)+C)’이며, (A’\*B’)+C’ = ((A+B)\*C)’ 이다.

**4. 1-bit 비교기**

**테이블이(가) 표시된 사진

자동 생성된 설명**



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A** | **B** | **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-bit 비교기는 진리표에서 보는 것처럼 = ,!=, >, < 을 사용해 비트를 비교하는 것이다.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **A** | **B** | **(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 |

그러나, 실습에서 결과가 동일하게 나오는 not, and, xor을 사용한 위의 스케메틱 회로를 사용했다. 진리표의 결과는 동일하다. 시뮬레이션은 의도한대로 결과가 출력 되었다. A == B와 A != B는 서로 not의 관계에 있다. 또한, A != B의 경우 xor과 동일하다. 더불어, A>B라는 것은 A는 1, B는 0 ( 이진수에는 1과 0 뿐이다.)을 의미하므로, A와 B’의 and 관계만 1을 출력할 것이다. 반대의 경우도 같은 방식으로 구현 가능하다.

**5. 결과 검토 및 논의**

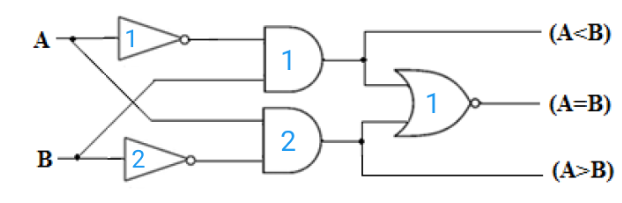
드모르간의 제1법칙과 제 2법칙의 결과는 각각 **NOR게이트와 NAND게이트만으로 회로를 구성**할 수 있음을 알려준다. NAND게이트와 NOR게이트는 가장 효율적인 게이트이므로 이는 굉장한 의의를 가진다. 또한, 복잡해 보이는 불 식을 간단하게 만들어주는 기능도 한다. 실습에서 연습한 Boolean 식은 드모르간의 응용으로 볼 수 있다. 결과적으로 드모르간의 법칙이 성립함을 보여주며, 이 역시 AND,OR게이트로 이루어진 SOP 식을 NAND와 NOR을 이용한 회로의 구성이 가능함을 보여준다.

1bit 비교기는 실제로 코드에서 y1 = (A == B)와 같은 방식으로 작성이 가능하다. (C언어와 유사한 문법을 가진다.) 이는 논리연산자의 사용이 가능함을 의미한다.

**6. 추가 이론**

1-bit 비교기를 이용해 n-bit 비교기의 구현도 가능하다. 앞서 배운 1-bit 비교기를 응용하면, (A < B) + (A > B) = A’B + AB’ 이 된다. 양변에 보수를 취해주면, 이는

((A < B) + (A > B))’ = (A == B) 이며, 우변은 AB + A’B’이 된다. 이를 통해 앞서 나온 스케메틱과 조금 다른 다음과 같은 회로를 구성 할 수 있다.



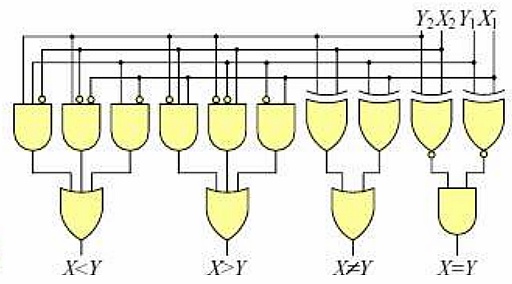
단일 비트 비교기를 2번 호출하여 2-bit 비교기를 구현할 수도 있지만, k-map을 사용해 구현도 가능하다. 다음 진리표를 통해, k-map을 사용하면,

테이블이(가) 표시된 사진

자동 생성된 설명텍스트, 낱말맞추기게임, 영수증이(가) 표시된 사진

자동 생성된 설명

이를 통해, 다음과 같은 2-bit 비교기 회로를 만들 수 있다.



**7. 참고문헌**

[**https://www.deldsim.com/study/material/14/one-bit-comparator/**](https://www.deldsim.com/study/material/14/one-bit-comparator/)

[**https://www.geeksforgeeks.org/magnitude-comparator-in-digital-logic/**](https://www.geeksforgeeks.org/magnitude-comparator-in-digital-logic/)