5주차 결과보고서

전공: 기계공학과 학년: 3학년 학번: 20191820 이름: 김형준

**1.**

▪ 드모르간의 정리와 Boolean 함수의 동작을 이해하고 확인해본다.

▪ Verilog를 사용하여 드모르간의 정리 및 Boolean 함수의 동작을 구현한다.

▪ 입력 신호를 생성한 다음 Simulation을 통하여 구현된 결과를 확인한다.

▪ FPGA를 사용해서 Verilog로 구현된 회로의 동작을 확인한다.

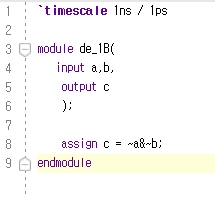
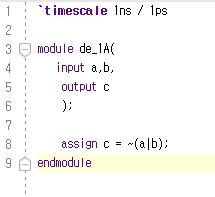
**2.**

- 드모르간의 제 1법칙

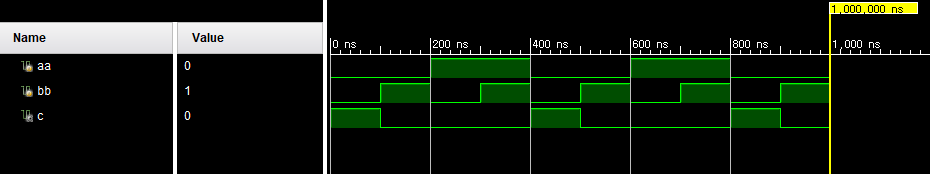
드모르간의 제 1법칙은 논리합의 부정은 각각의 부정을 논리곱 한 것과 같다는 법칙이다.

이를 Boolean 식으로 나타내면 다음과 같다.

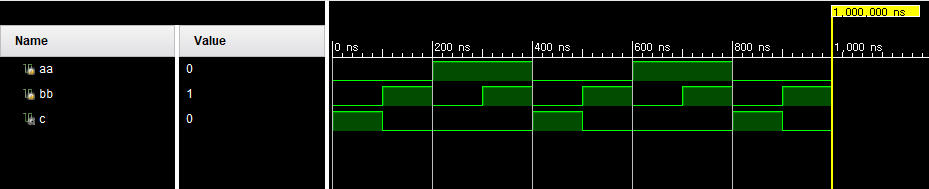
양변의 수식을 각각 Verilog상에서 구현하면 다음과 같다.



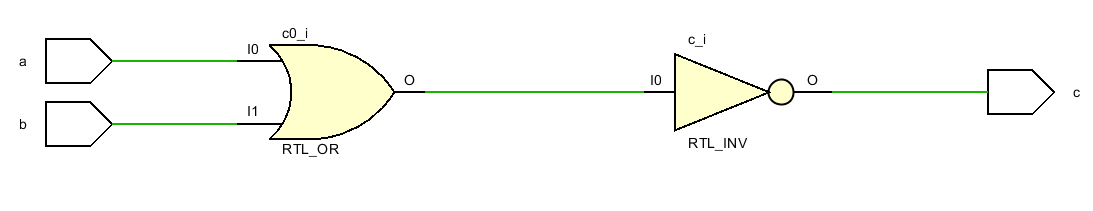
[드모르간의 제 1법칙을 구현한 Verilog 코드 (좌 : 좌변, 우 : 우변)]



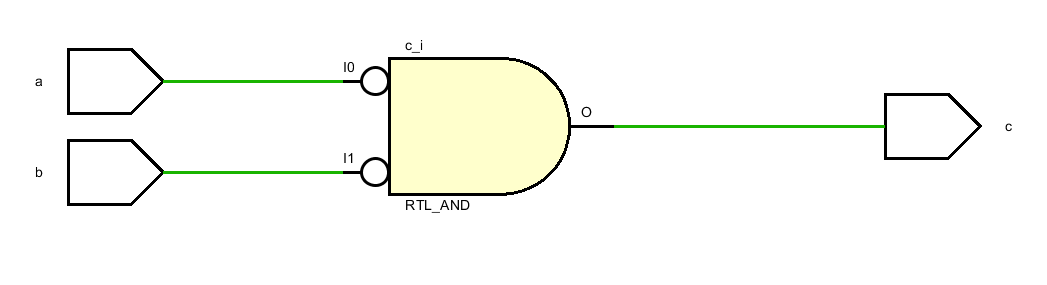
[드모르간의 제 1법칙의 Simulation 결과 (좌변)]



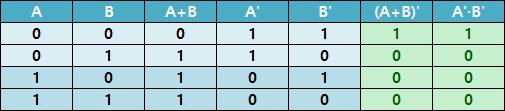
[드모르간의 제 1법칙의 Simulation 결과 (우변)]



[드모르간의 제 1법칙의 Schematic (좌변)]



[드모르간의 제 1법칙의 Schematic (우변)]



[드모르간의 제 1법칙의 진리표]

위의 드모르간 제 1법칙의 식을 보면, 좌변은 두 변수의 합에 부정을 취하고 있고, 우변은 두 변수 각각의 부정의 곱을 취하고 있다. 좌변의 경우 NOR 게이트의 정의(와 같으므로 A, B가 모두 0일때만 결과값으로 1을 갖고, 나머지의 경우에는 0을 갖는다. (드모르간 제 1법칙의 좌변의 Schematic이 NOR 게이트와 일치함을 알 수 있다.)

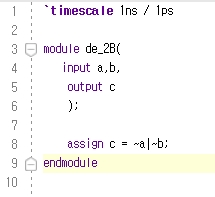
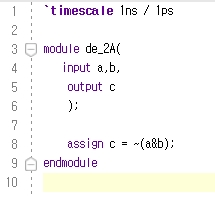
우변은 A’와 B’의 AND연산의 결과이다(. 따라서 A’와 B’가 모두 1일때만 결과값으로 1을 가지므로 A와 B가 모두 0일때만 결과값으로 1을, 나머지의 경우에는 0을 갖게 된다. 따라서 결과적으로 양변의 수식은 A와 B의 모든 입력에 대해 동일한 결과값을 갖는다. 이와 같이 드모르간 1법칙의 양변이 같다는 사실은 위의 진리표(초록색 부분)에서 다시 한번 확인할 수 있다.

- 드모르간의 제 2법칙

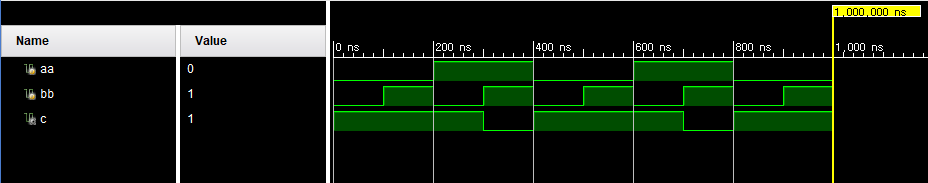
드모르간의 제 2법칙은 논리곱의 부정은 각각의 부정을 논리합 한 것과 같다는 법칙이다.

이를 Boolean 식으로 나타내면 다음과 같다.

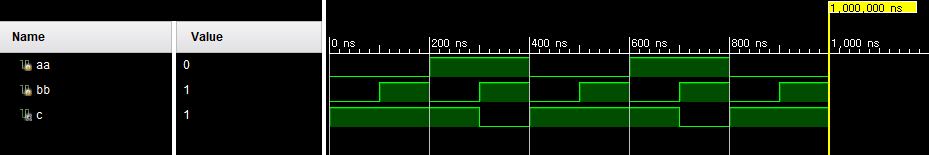
양변의 수식을 각각 Verilog상에서 구현하면 다음과 같다.



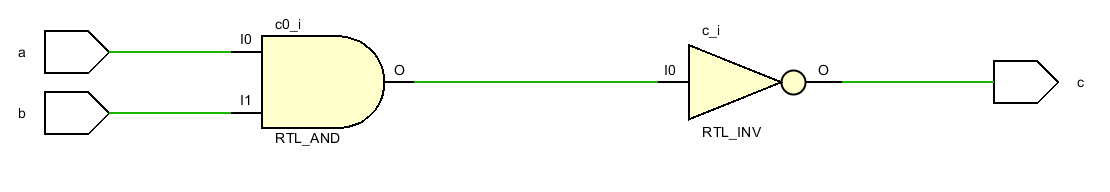
[드모르간의 제 2법칙을 구현한 Verilog 코드 (좌 : 좌변, 우 : 우변)]



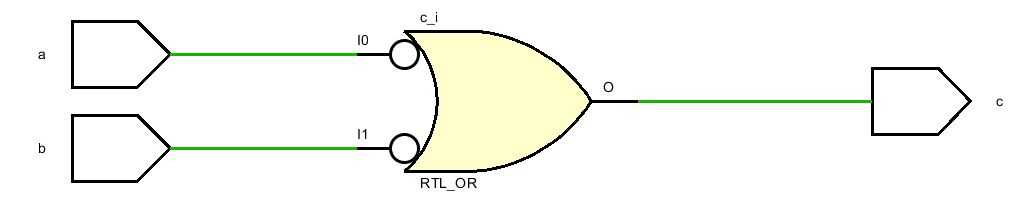
[드모르간의 제 2법칙의 Simulation 결과 (좌변)]



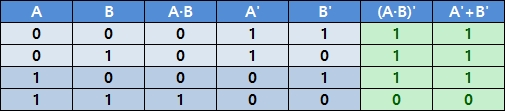
[드모르간의 제 2법칙의 Simulation 결과 (우변)]



[드모르간의 제 2법칙의 Schematic (좌변)]



[드모르간의 제 2법칙의 Schematic (우변)]



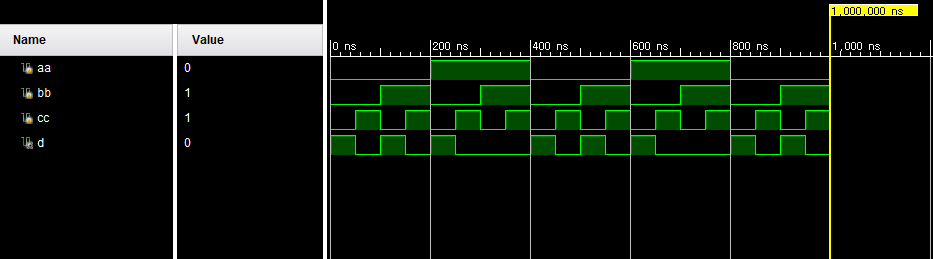
[드모르간의 제 2법칙의 진리표]

위의 드모르간 제 2법칙의 식을 보면, 좌변은 두 변수의 곱에 부정을 취하고 있고, 우변은 두 변수 각각의 부정의 합을 취하고 있다. 좌변의 경우 NAND 게이트의 정의(와 같으므로 A, B가 모두 1일때만 결과값으로 0을 갖고, 나머지의 경우에는 1을 갖는다. (드모르간 제 2법칙의 좌변의 Schematic이 NAND 게이트와 일치함을 알 수 있다.)

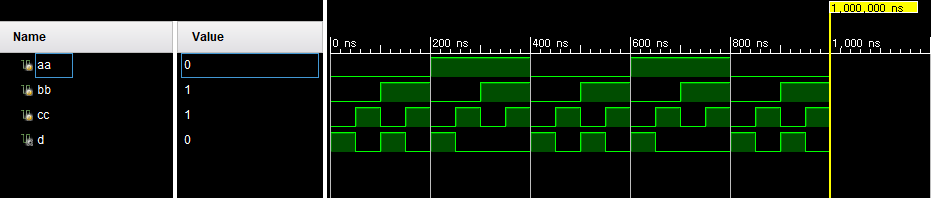
우변은 A’와 B’의 OR연산의 결과이다(. 따라서 A’와 B’가 모두 0일때만 결과값으로 0을 가지므로 A와 B가 모두 1일때만 결과값으로 0을, 나머지의 경우에는 1을 갖게 된다. 따라서 결과적으로 양변의 수식은 A와 B의 모든 입력에 대해 동일한 결과값을 갖는다. 이와 같이 드모르간 2법칙의 양변이 같다는 사실은 위의 진리표(초록색 부분)에서 다시 한번 확인할 수 있다.

**3.**

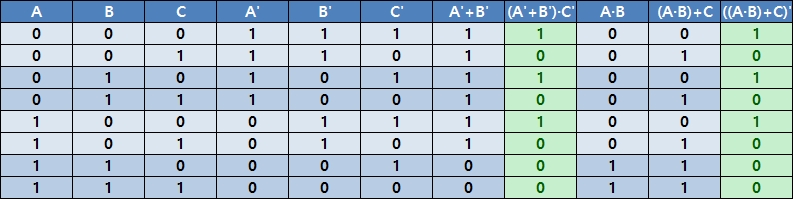
- Boolean Function 1번 : (



[좌변((의 Simulation 결과]



[우변(의 Simulation 결과]



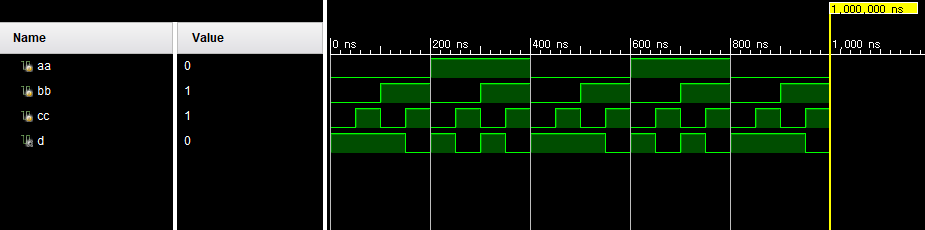
[Boolean Function((의 진리표]

우변의 수식을 드모르간의 법칙을 사용해 좌변과 같음을 아래와 같은 방식으로 증명할 수 있다.

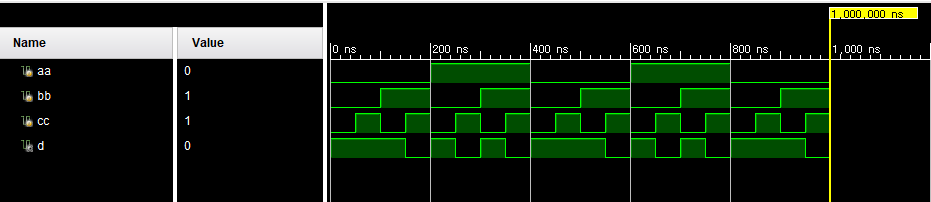
이는 위의 진리표(초록색 부분)에서 양변이 같다는 사실을 다시 한번 확인할 수 있다.

좌변(()이 1이 되는 경우를 분석해보면, 좌변의 결과값은 와 의 AND연산의 결과이므로 와 모두 1이여야만 한다. 따라서 C는 0이여야 하고, 가 1이 되는 조건을 찾아보면, 는 A’와 B’의 OR연산의 결과이므로 A’와 B’가 모두 0이 아닌 경우를 제외하면 된다. 따라서 (()가 1이 되는 경우는 C가 0인 경우 중 A와 B가 모두 1인 경우만 제외한 경우이다. 이에 해당하는 A, B, C의 입력은 (0,0,0), (0,1,0), (1,0,0)이 있고, 나머지의 경우는 모두 0을 결과값으로 가지게 된다. 위에서 양변이 항상 같음을 보였으므로 Boolean 함수 1번은 위 진리표와 같은 진리값(초록색 부분)을 가지게 된다.

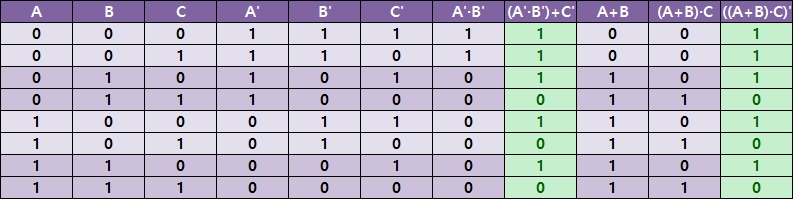
- Boolean Function 2번 : (



[좌변 ((의 Simulation 결과]



[우변()의 Simulation 결과]



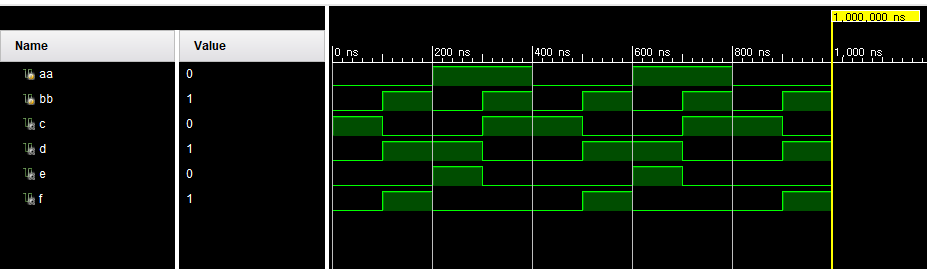
[Boolean Function (()의 진리표]

우변의 수식을 드모르간의 법칙을 사용해 좌변과 같음을 아래와 같은 방식으로 증명할 수 있다.

이는 위의 진리표(초록색 부분)에서 양변이 같다는 사실을 다시 한번 확인할 수 있다.

좌변(()이 0이 되는 경우를 분석해보면, 좌변의 결과값은 와 의 OR연산의 결과이므로 와 모두 0이여야만 한다. 따라서 C는 1이여야 하고, 가 0이 되는 조건을 찾아보면, 는 A’와 B’의 AND연산의 결과이므로 A’와 B’가 모두 1인 경우를 제외하면 된다. 따라서 (()가 0이 되는 경우는 C가 1인 경우 중 A와 B가 모두 0인 경우만 제외한 경우이다. 이에 해당하는 A, B, C의 입력은 (0,1,1), (1,0,1), (1,1,1)이 있고, 나머지의 경우는 모두 1을 결과값으로 가지게 된다. 위에서 양변이 항상 같음을 보였으므로 Boolean 함수 2번은 위 진리표와 같은 진리값(초록색 부분)을 가지게 된다.

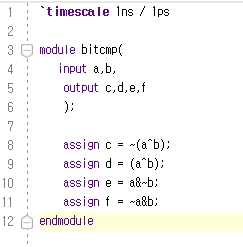
**4.**



[1Bit 비교기의 Simulation 결과]



[1Bit 비교기의 진리표]



[1Bit 비교기를 구현한 Verilog 코드]

의 경우 A와 B가 서로 같을 때 1, 다를 때는 0을 반환해야 한다. 이는 XNOR 게이트의 진리값과 동일하므로 를 로 구현할 수 있다. 반대로, 의 경우에는 A와 B가 서로 같을 때 0, 다를 때 1을 반환해야 하는데, 이는 XOR 게이트의 진리값과 동일하므로 를 로 구현할 수 있다. 의 경우 모든 A, B의 입력중 를 만족하는 입력은 (1,0) 뿐이므로 (1,0)에서만 1을, 나머지 경우에는 0을 반환해야 한다.

이는 로 구현할 수 있는데, 가 1이 되는 경우는 A와 B’가 모두 1인 경우에만 해당하므로 이를 찾으면 (1,0) 뿐이다. 이는 의 진리값과 동일하므로 로 를 구현할 수 있다. 의 경우 마찬가지로 모든 A, B의 입력중 를 만족하는 입력은 (0,1) 뿐이므로 (0,1)에서만 1을, 나머지 경우에는 0을 반환해야 한다.

이는 로 구현할 수 있는데, 가 1이 되는 경우는 A’와 B가 모두 1인 경우에만 해당하므로 이를 찾으면 (0,1) 뿐이다. 이는 의 진리값과 동일하므로 로 를 구현할 수 있다.

**5.**

3.과 4.에서 각각의 시뮬레이션 결과와 진리표를 비교했을 때 차이가 없으므로 의도한 대로

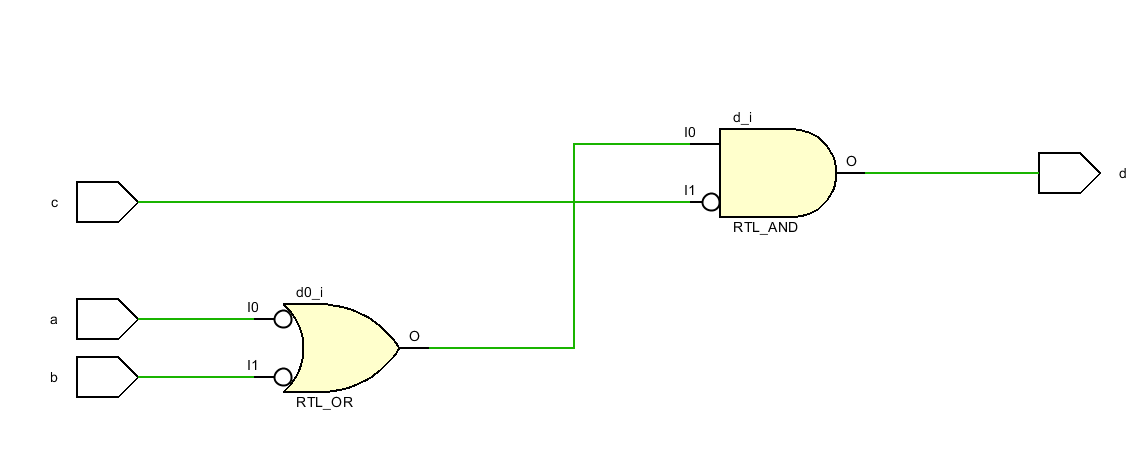
코드가 작성되었음을 확인 할 수 있다. 이 실험을 통해 드모르간 제 1, 2 법칙이 성립하고,

각각 NOR, NAND 게이트와 같은 진리값을 가짐을 확인할 수 있었다.

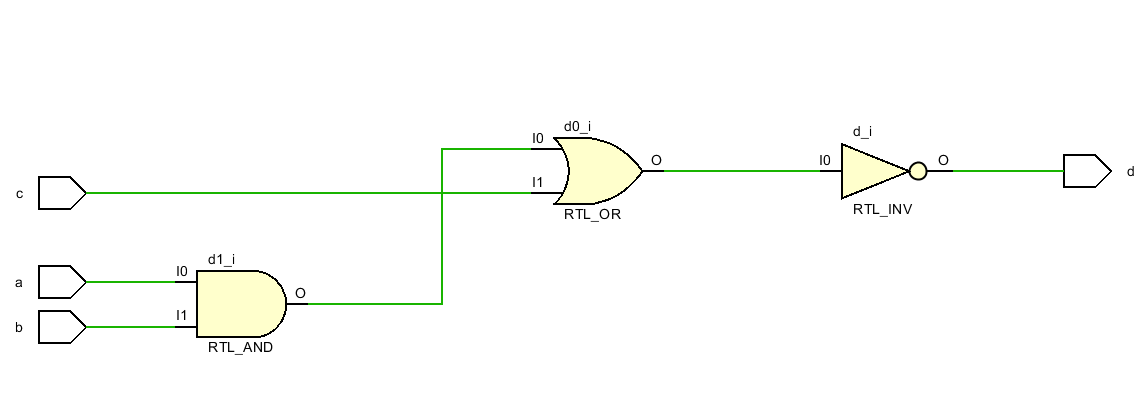
또한, 복잡한 Boolean 수식을 드모르간 법칙을 여러 번 사용하여 보수를 취할 수 있음을 알 수 있었다.

1Bit 비교기의 구현을 통해 각 비트를 비교하는 연산을 기본 비트 연산자로 구현하는 방법에 대해서 알아볼 수 있었다.

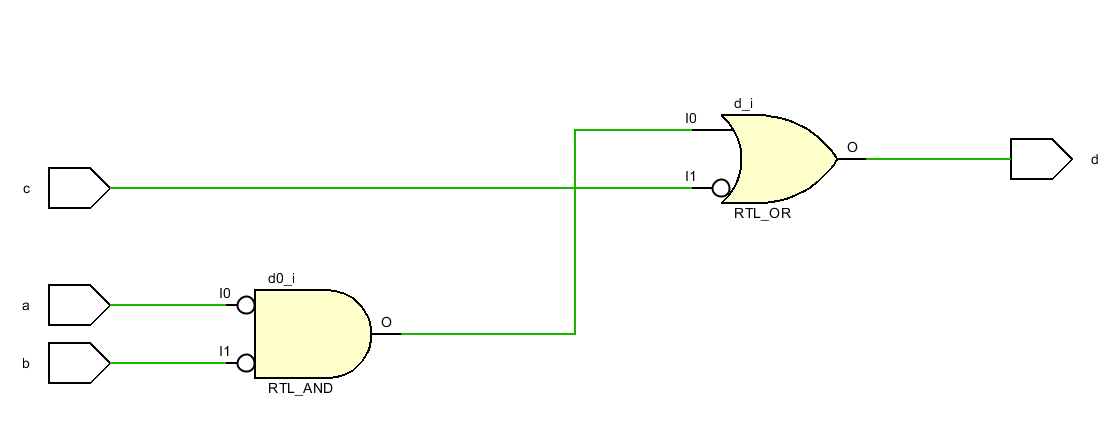
아래의 사진들은 3. ~ 4. 각각의 Schematic이다.



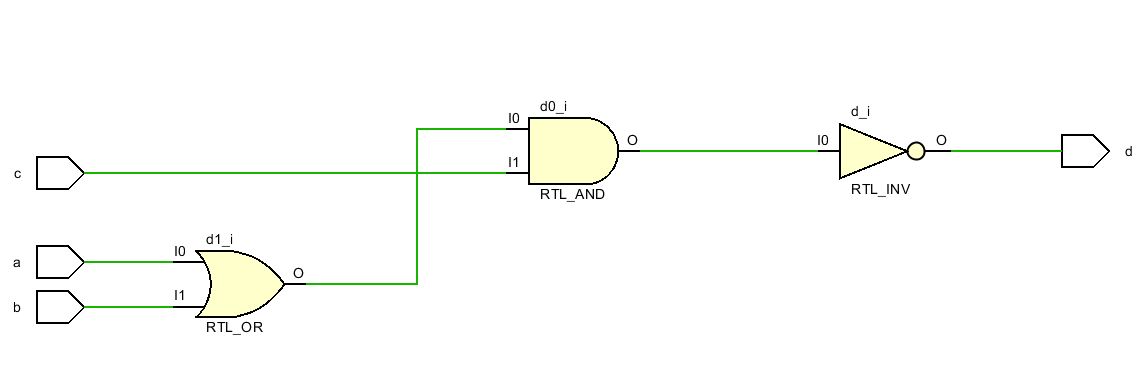
[Boolean 함수 1번 : 좌변((의 Schematic]



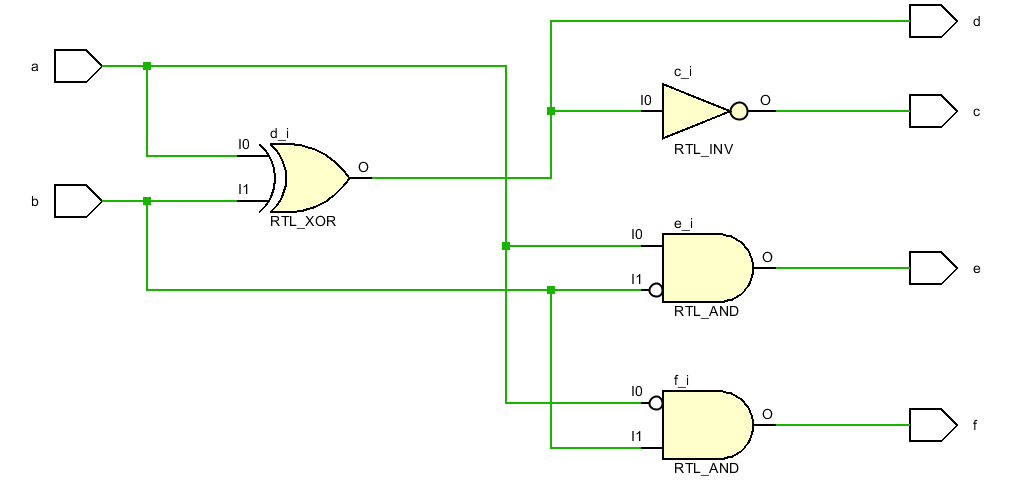
[Boolean 함수 1번 : 우변()의 Schematic]



[Boolean 함수 2번 : 좌변 ((의 Schematic]



[Boolean 함수 2번 : 우변()의 Schematic]



[1Bit 비교기의 Schematic]

**6.**

3개 이상의 피연산자를 갖는 수식에 대해서도 드모르간의 법칙을 적용할 수 있게 확장 할 수 있다. 아래와 같은 방법으로 드모르간의 법칙을 변수가 3개인 경우로 확장시킬 수 있다.

위와 같은 방법으로 n개의 변수에 대해 드모르간의 법칙을 확장시키면, 다음과 같은 공식을 얻을 수 있다.

복잡한 Boolean 수식의 보수를 구하는 또 다른 방법으로, 아래와 같은 규칙을 반복해서 적용하여 구할 수 있다.

1. 각 변수를 보수화 한다. (를 로, 로 교체)

2. 0과 1을 각각 1과 0으로 바꾼다.

3. AND는 OR로, OR은 AND로 바꾸고 연산의 순서가 보존될 수 있도록 괄호를 추가한다.

예를들어, 의 보수를 위의 규칙을 사용해 구하면,

이 된다.