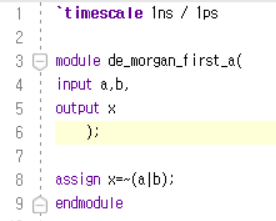
5주차 결과보고서

전공: 심리학과 학년: 3학년 학번: 20190345 이름: 김동현

**1.**

드 모르간의 정리, Boolean function, 1비트 비교기를 베릴로그를 이용하여 구현한다. 시뮬레이션을 통해 구현한 결과를 확인하고, FPGA를 통해 시뮬레이션 결과와 비교한다.

2.



<사진1. x=~(a|b)의 design code>

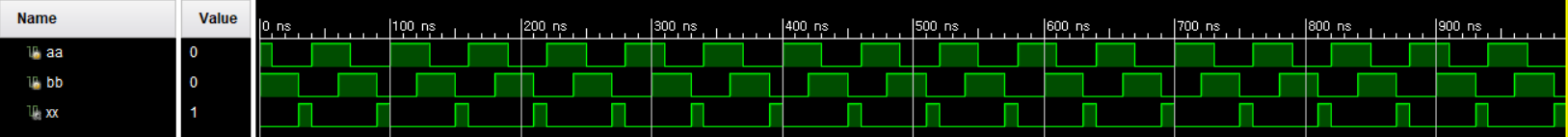
위 사진은 x=~(a|b)을 구현한 design source code이다. 입력 변수 a,b와 출력변수 x를 선언하고, x에 ~(a|b)를 할당하였다.

테이블이(가) 표시된 사진

자동 생성된 설명

<사진2. x=~(a|b)의 simulation code>

X=~(a|b)를 시뮬레이션으로 나타내기 위해 simulation source code를 작성하였다.



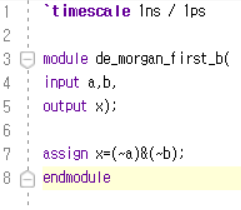
<사진3. x=~(a|b)의 simulation>

X=~(a|b)를 시뮬레이션으로 나타내었다.



<사진4. x=~(a|b)의 schmetics>

X=~(a|b)를 schematic으로 나타내었다.



<사진5. x=(~a)&(~b)의 design code>

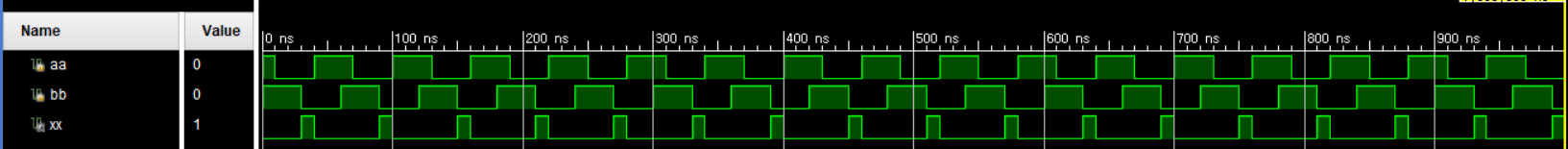
위 사진은 x=(~a)&(~b)을 구현한 design source code이다. 입력 변수 a,b와 출력변수 x를 선언하고, x에 (~a)&(~b)를 할당하였다.

텍스트이(가) 표시된 사진

자동 생성된 설명

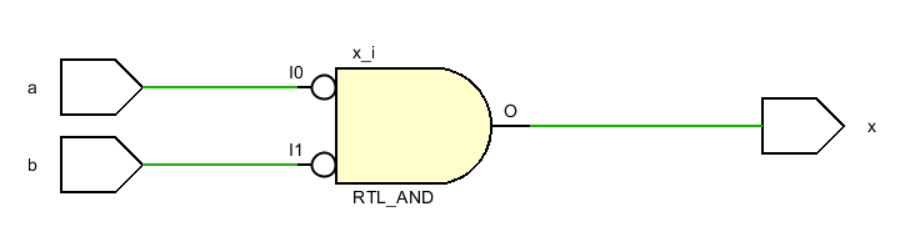
<사진6. x=(~a)&(~b)의 simulation code>

x=(~a)&(~b)를 시뮬레이션으로 나타내기 위해 simulation source code를 작성하였다.



<사진7. x=(~a)&(~b)의 simulation>

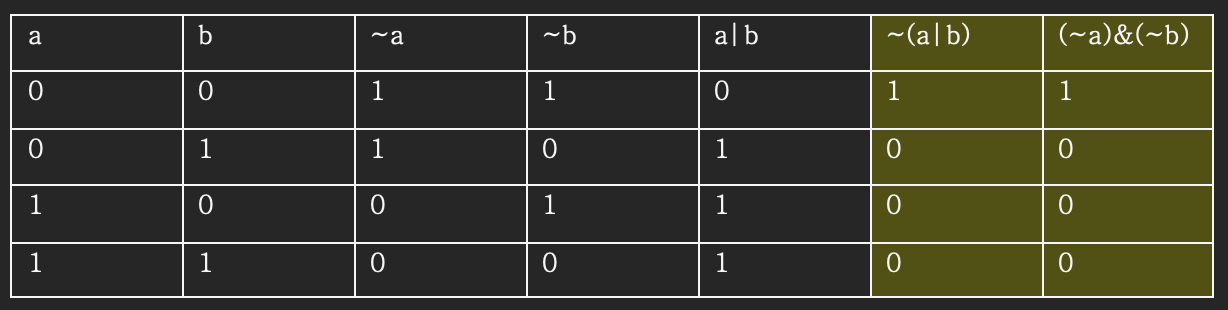
x=(~a)&(~b)를 시뮬레이션으로 나타내었다.



<사진8. x=(~a)&(~b)의 schematics>

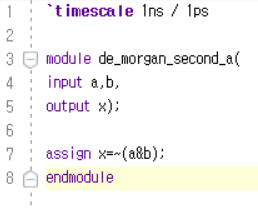
x=(~a)&(~b)를 schematic으로 나타내었다.

~(a|b)=(~a)&(~b)이 같다는 것을 보이기 위해 진리표를 작성하였다.



<표1. ~(a|b), (~a)&(~b) 진리값 비교>

표를 보면 알 수 있듯이 모든 경우의 수에 대해 같은 결과 값을 보이고 있다. 즉, ~(a|b) = (~a) & (~b)임을 보이고 있고, 이것이 드 모르간의 제2법칙이다.



<사진9. x=~(a&b)의 design code>

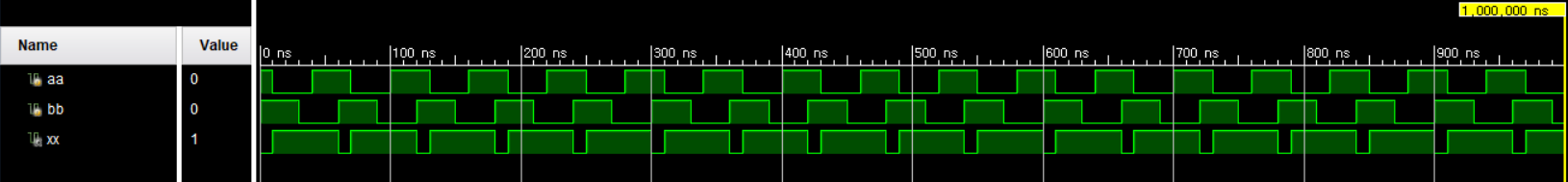
위 사진은 x=~(a&b)을 구현한 design source code이다. 입력 변수 a,b와 출력변수 x를 선언하고, x에 ~(a&b)를 할당하였다.

테이블이(가) 표시된 사진

자동 생성된 설명

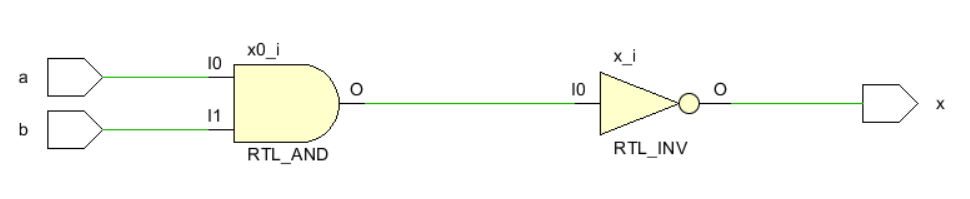
<사진10. x=~(a&b)의 simulation code>

X=~(a&b)를 시뮬레이션으로 나타내기 위해 simulation source code를 작성하였다.



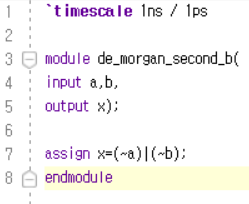
<사진11. x=~(a&b)의 simulation>

X=~(a&b)를 시뮬레이션으로 나타내었다.



<사진12. x=~(a&b)의 schematics>

X=~(a&b)를 schematic으로 나타내었다.



<사진13. x=(~a)|(~b)의 design code>

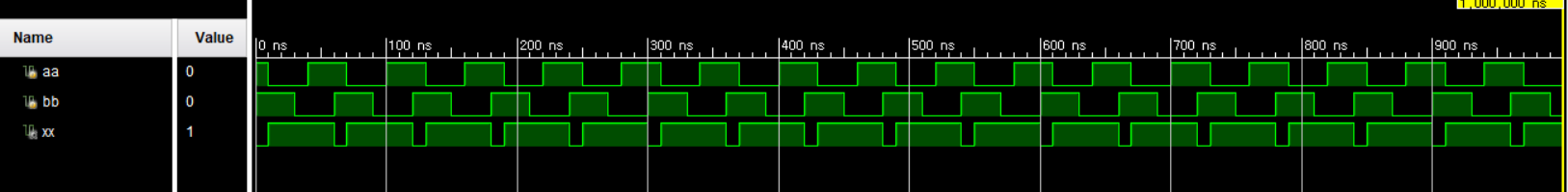
위 사진은 x=(~a)|(~b)을 구현한 design source code이다. 입력 변수 a,b와 출력변수 x를 선언하고, x에 (~a)|(~b)를 할당하였다.

테이블이(가) 표시된 사진

자동 생성된 설명

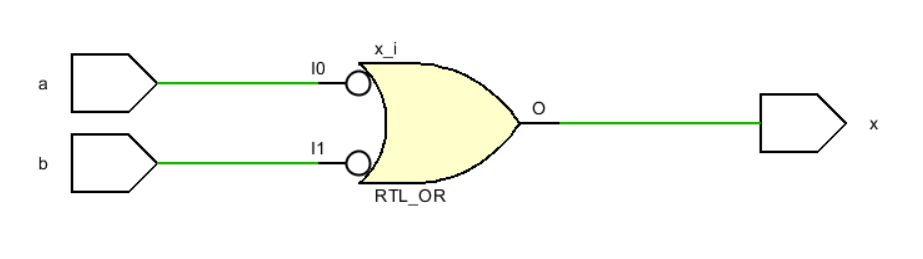
<사진14. x=(~a)|(~b)의 simulation code>

x=(~a)|(~b)를 시뮬레이션으로 나타내기 위해 simulation source code를 작성하였다.



<사진15. x=(~a)|(~b)의 simulation>

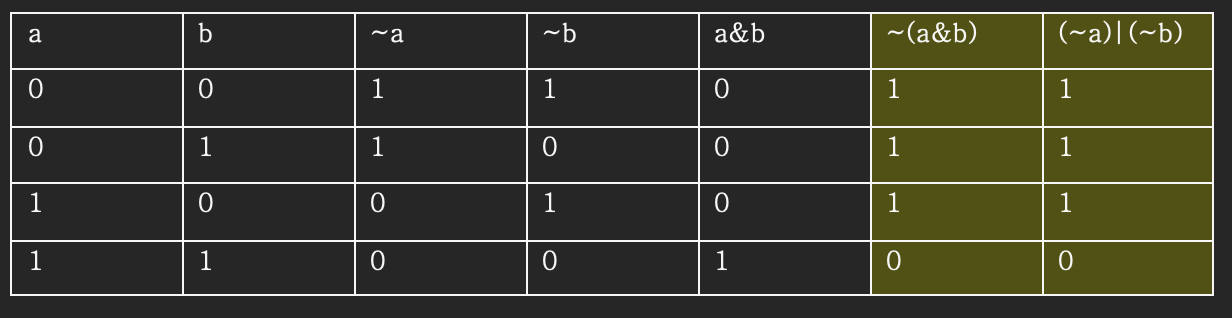
x=(~a)|(~b)를 시뮬레이션으로 나타내었다.



<사진16. x=(~a)|(~b)의 schematics>

x=(~a)|(~b)를 schematic으로 나타내었다.

~(a&b)=(~a)|(~b)을 보이기 위해 진리표를 작성하였다.

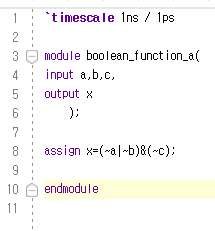


<표2. ~(a&b), (~a)|(~b) 진리값 비교>

위 표에서 알 수 있듯이 모든 경우의 수에 대해 ~(a&b)=(~a)|(~b)를 만족하고 있다. 이는 드 모르간의 제1법칙이다.

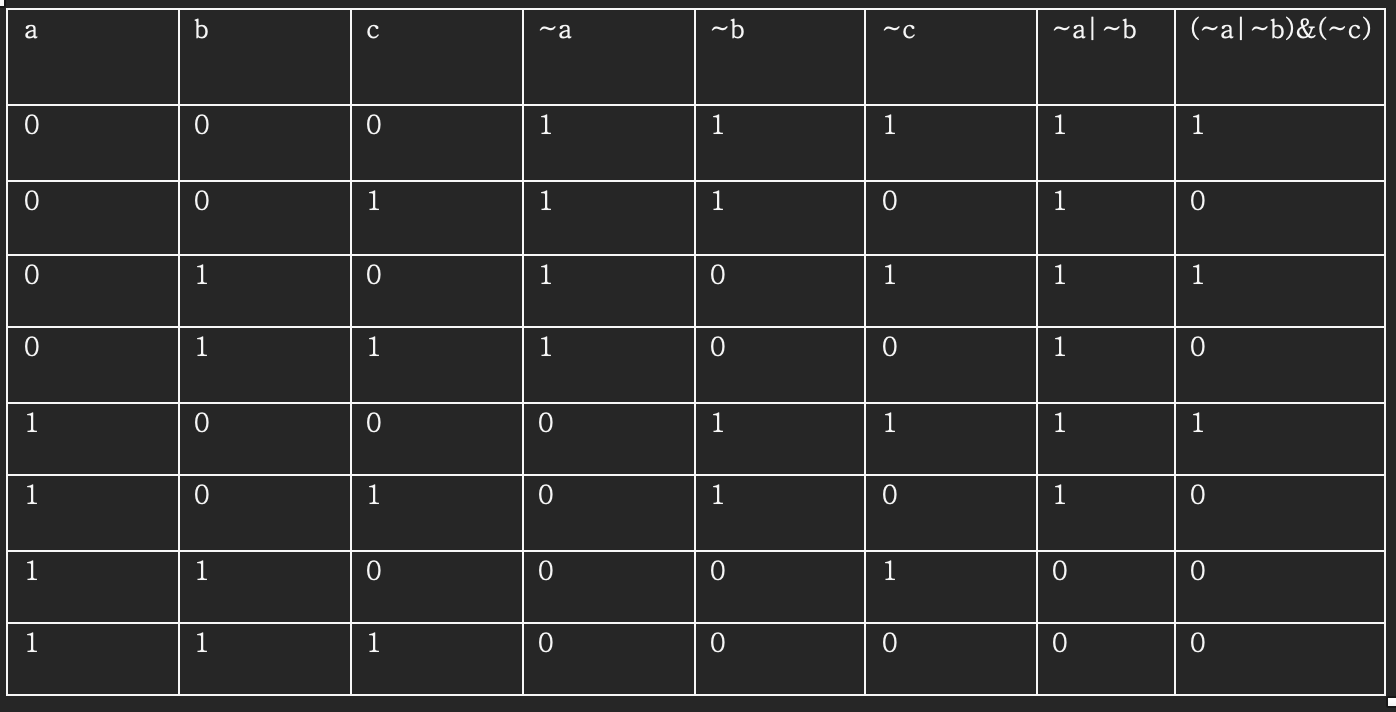
3.

(A’+B’) \* C’을 베릴로그로 구현한 코드 및 결과값이다.



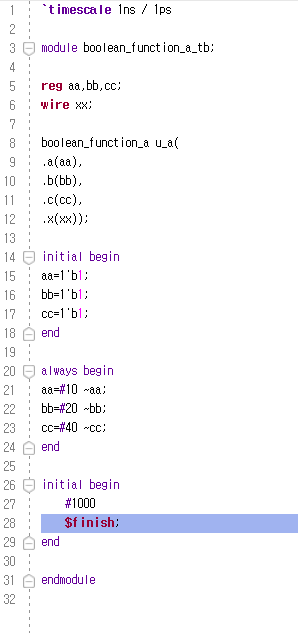
<사진17. (~a|~b)&(~c)의 design source>

위 사진은 (~a|~b)&(~c)에 대한 디자인 소스 파일이다. 입력값으로 a,b,c를 선언하고, 출력값으로 x를 선언하였다. 이후 출력값 x에 (~a|~b)&(~c)을 할당하였다.



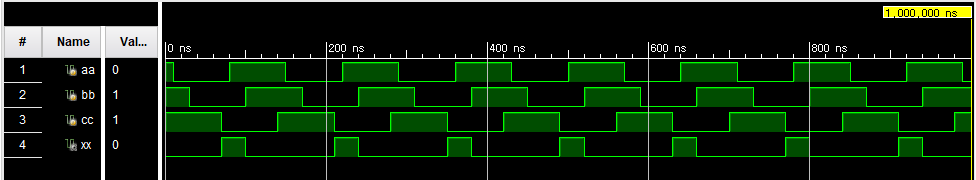
<표3. (~a|~b)&(~c)의 truth table >

(~a|~b)&(~c)에 대한 결과값을 진리표로 나타내었다.



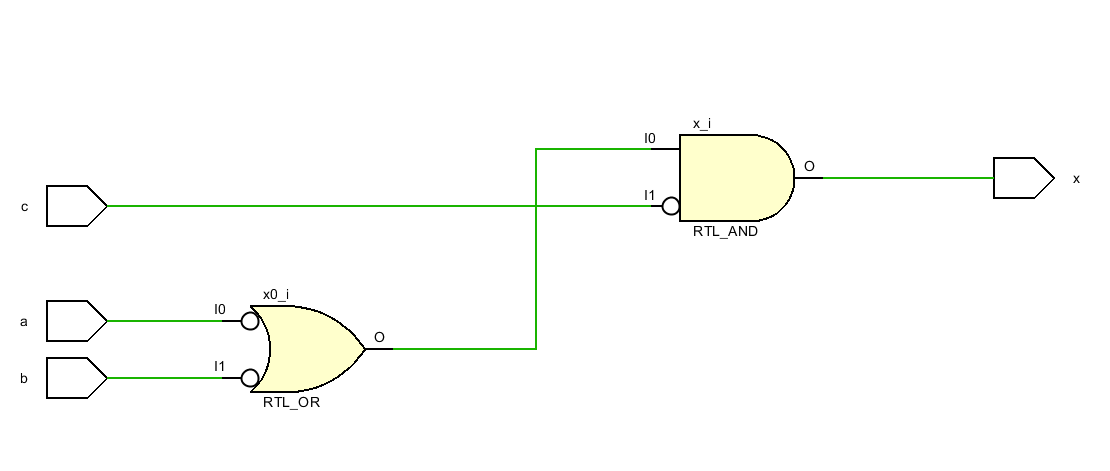
<사진18. (~a|~b)&(~c)의 simulation source>

(~a|~b)&(~c)를 시뮬레이션으로 구현하기 위한 시뮬레이션 코드이다.



<사진19. (~a|~b)&(~c)의 simulation>

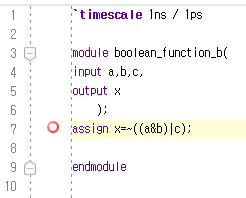
시뮬레이션 코드를 run하여 시뮬레이션을 구현하였다. 위에서 조사한 진리표와 비교해본다면 동일한 결과값을 갖는다는 것을 확인할 수 있다.



<사진20. (~a|~b)&(~c)의 schematics >

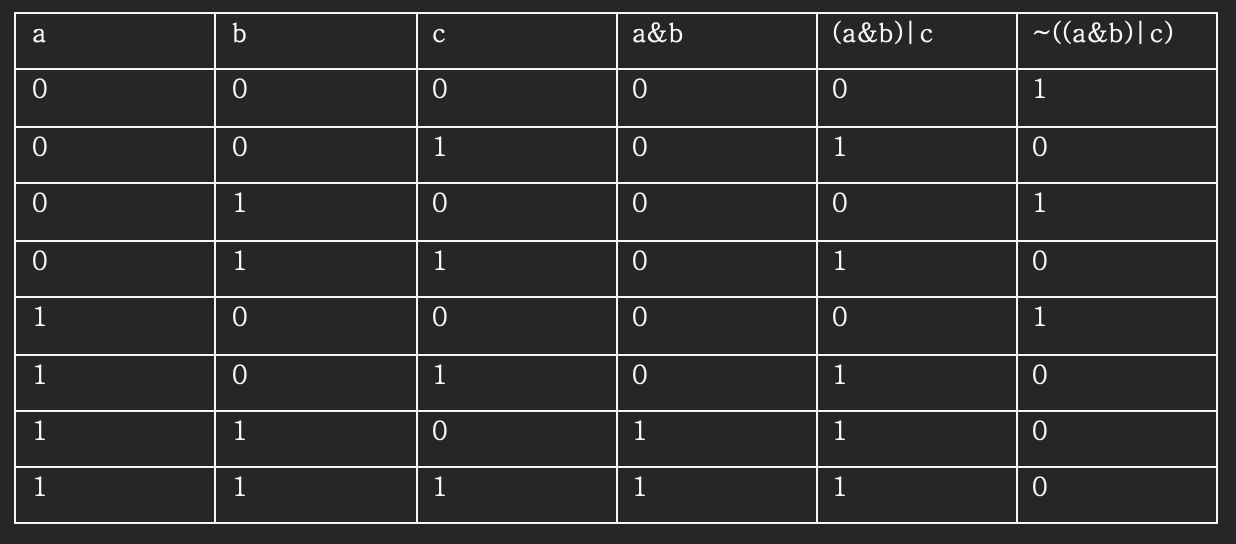
(~a|~b)&(~c)을 schematic으로 나타내었다.

((A\*B)+C)’을 베릴로그로 구현한 코드 및 결과이다.



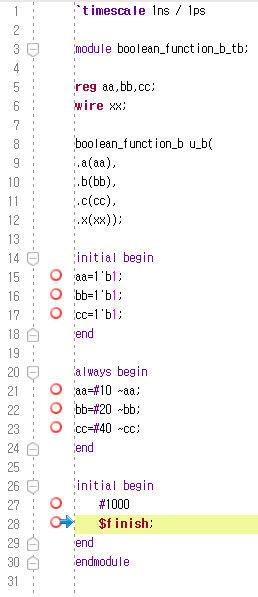
<사진21. ~((a&b)|c)의 design source>

위 사진은 ~((a&b)|c)에 대한 디자인 소스 파일이다. 입력값으로 a,b,c를 선언하고, 출력값으로 x를 선언하였다. 이후 출력값 x에 ~((a&b)|c)을 할당하였다.



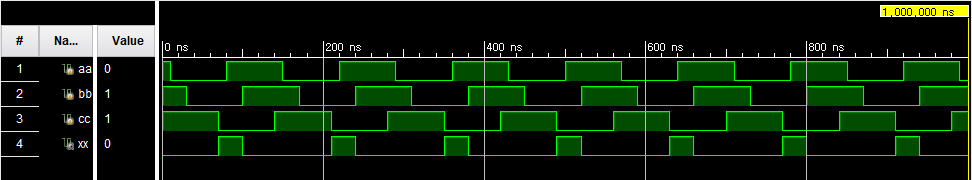
<표4. ~((a&b)|c)의 truth table >

~((a&b)|c) 에 대한 결과값을 진리표로 나타내었다.



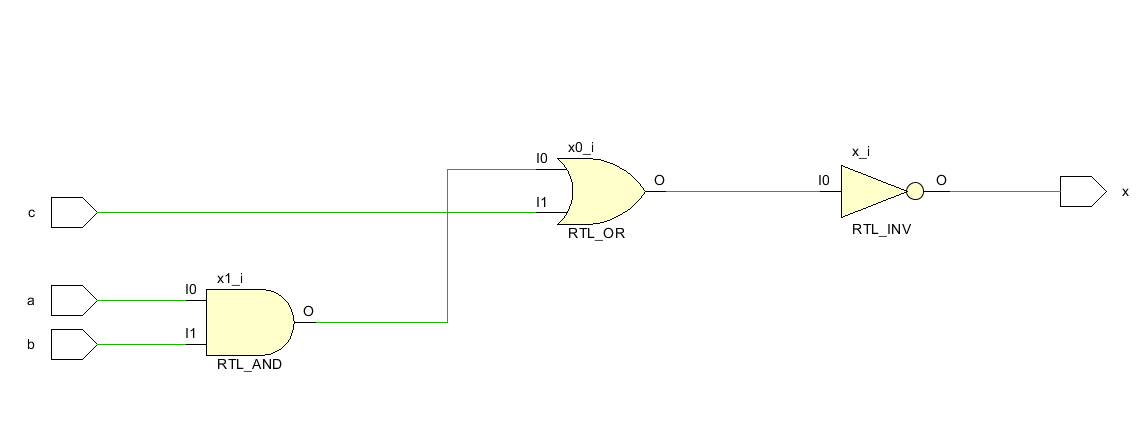
<사진22. ~((a&b)|c)의 simulation source>

~((a&b)|c) 를 시뮬레이션으로 구현하기 위한 시뮬레이션 코드이다.



<사진23. ~((a&b)|c)의 simulation>

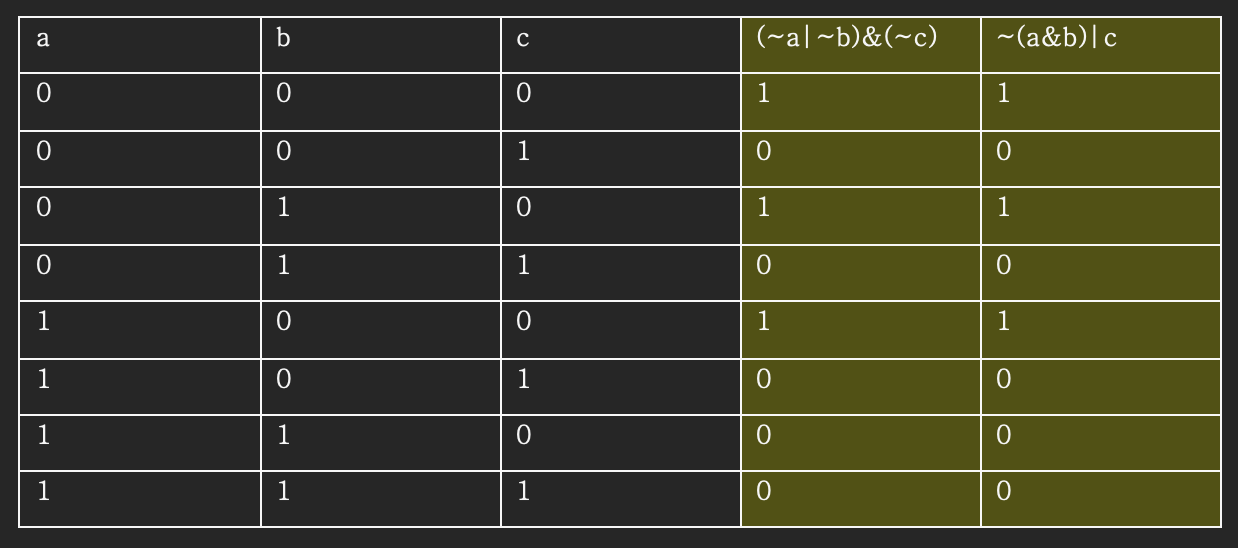
시뮬레이션 코드를 run하여 시뮬레이션을 구현하였다. 위에서 조사한 진리표와 비교해본다면 동일한 결과값을 갖는다는 것을 확인할 수 있다.



<사진24. ~((a&b)|c)의 schematics>

~((a&b)|c) 을 schematic으로 나타내었다.

(~a|~b)&(~c)와 ~(a&b)|c의 결과값을 비교하면 다음과 같다.



<표5. (~a|~b)&(~c)와 ~(a&b)|c의 진리표>

위의 결과값을 비교한다면, 입력값 a,b,c에 따른 출력값이 같은 것을 알 수 있다. 따라서 (~a|~b)&(~c) = ~((a&b)|c)임을 알 수 있다.

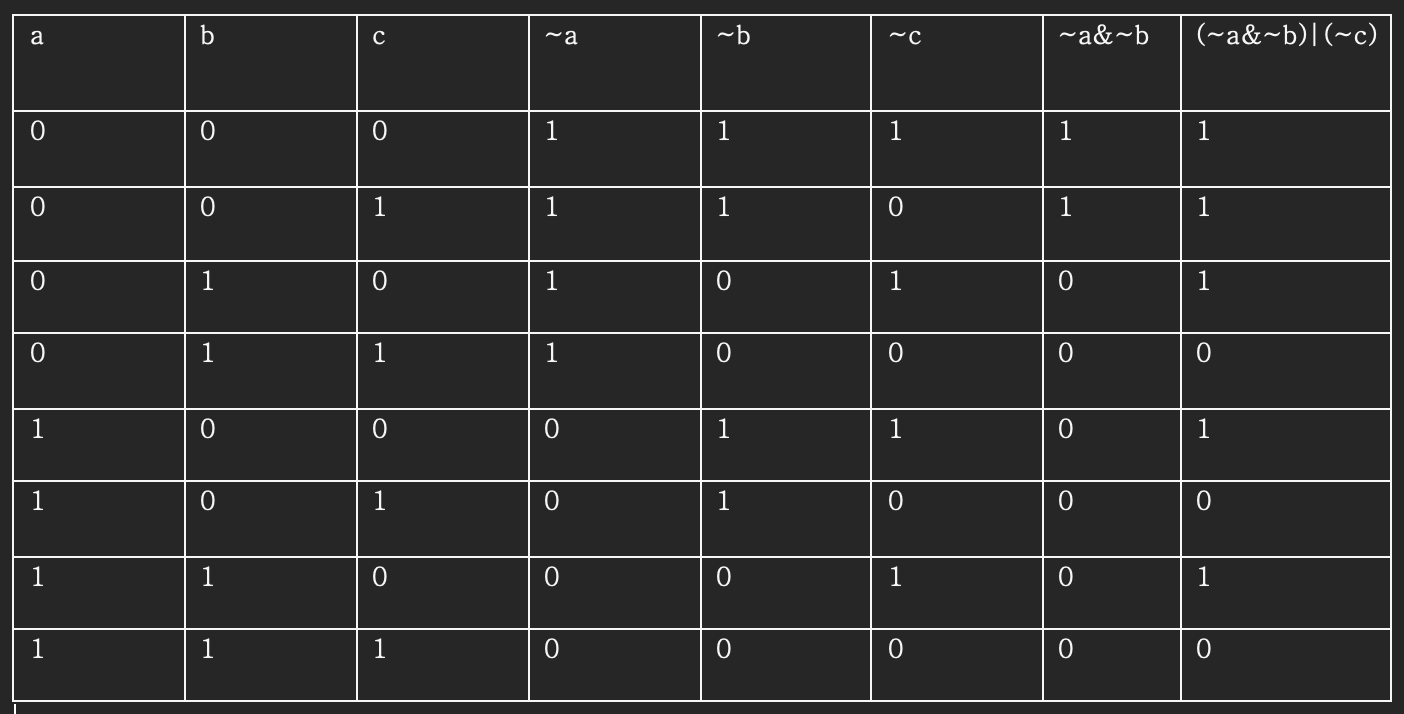
(A’\*B’)+C’ 을 베릴로그로 구현한 코드 및 결과값이다.

테이블이(가) 표시된 사진

자동 생성된 설명

<사진25. ((~a)&(~b))|(~c)의 design source>

위 사진은 ((~a)&(~b))|(~c)에 대한 디자인 소스 파일이다. 입력값으로 a,b,c를 선언하고, 출력값으로 x를 선언하였다. 이후 출력값 x에 ((~a)&(~b))|(~c)을 할당하였다.



<표6. ((~a)&(~b))|(~c)의 truth table>

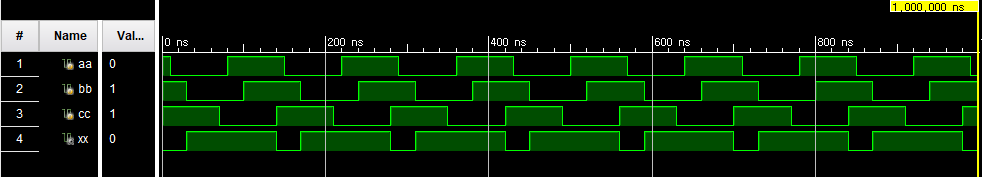
((~a)&(~b))|(~c)에 대한 결과값을 진리표로 나타내었다.

테이블이(가) 표시된 사진

자동 생성된 설명

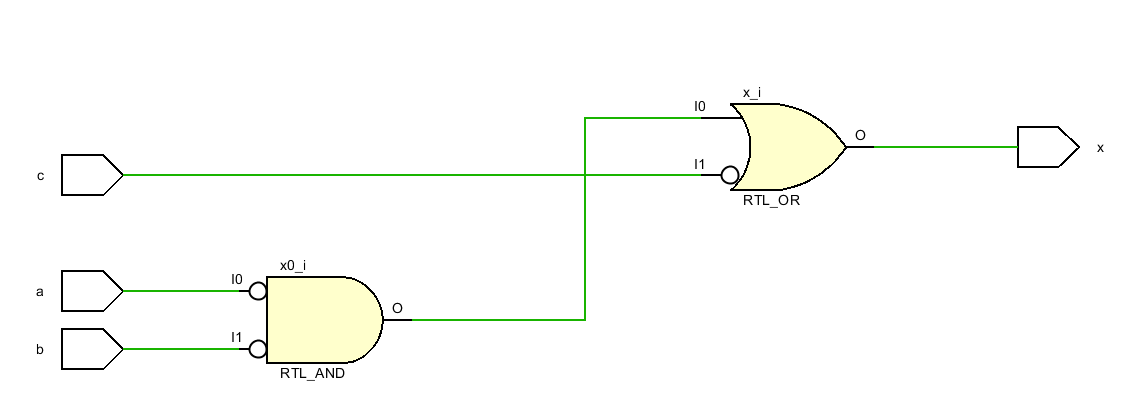
<사진26. ((~a)&(~b))|(~c)의 simulation source>

((~a)&(~b))|(~c)를 시뮬레이션으로 구현하기 위한 시뮬레이션 코드이다.



<사진27. ((~a)&(~b))|(~c)의 simulation>

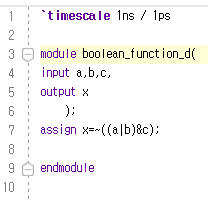
시뮬레이션 코드를 run하여 시뮬레이션을 구현하였다. 위에서 조사한 진리표와 비교해본다면 동일한 결과값을 갖는다는 것을 확인할 수 있다.



<사진28. ((~a)&(~b))|(~c)의 schematics>

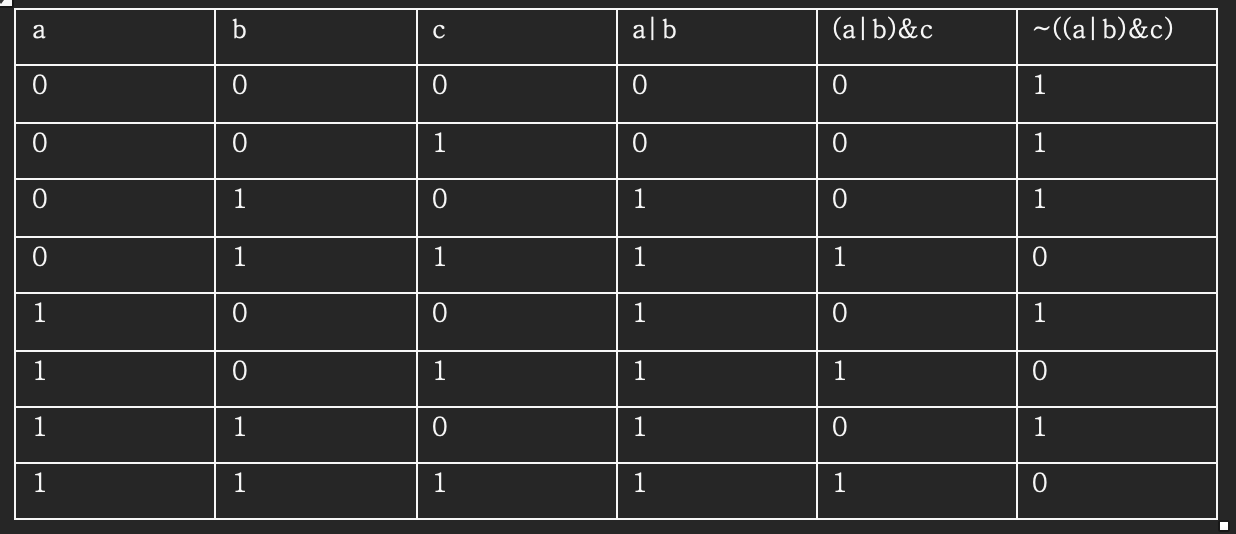
((~a)&(~b))|(~c)을 schematic으로 나타내었다.

((A+B)\*C)’ 을 베릴로그로 구현한 코드 및 결과값이다.



<사진29. ~((a|b)&c)의 design source>

위 사진은 ~((a|b)&c) 에 대한 디자인 소스 파일이다. 입력값으로 a,b,c를 선언하고, 출력값으로 x를 선언하였다. 이후 출력값 x에 ~((a|b)&c) 을 할당하였다.



<표7. ~((a|b)&c)의 truth table>

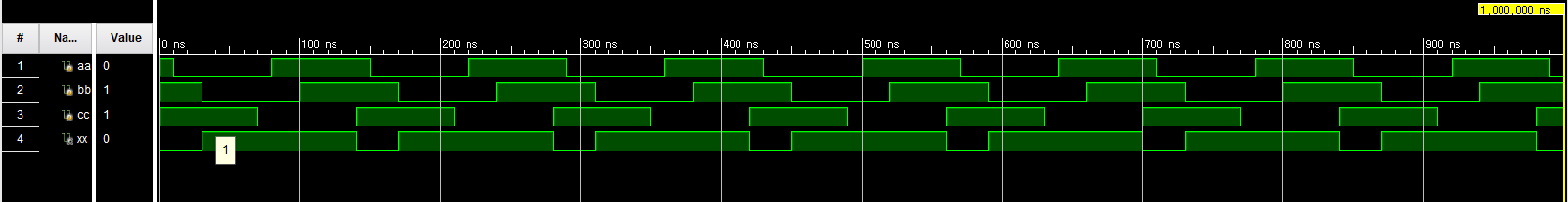
~((a|b)&c) 에 대한 결과값을 진리표로 나타내었다.

테이블이(가) 표시된 사진

자동 생성된 설명

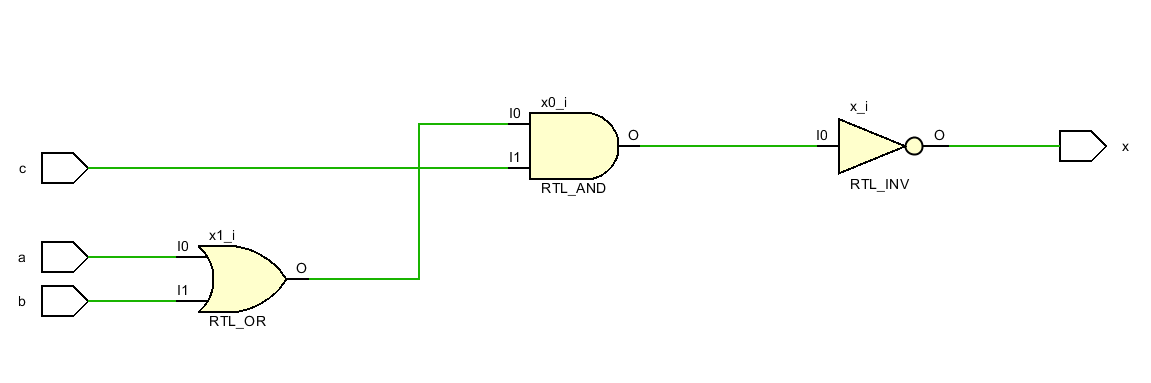
<사진30. ~((a|b)&c)의 simulation source>

~((a|b)&c) 를 시뮬레이션으로 구현하기 위한 시뮬레이션 코드이다.



<사진31. ~((a|b)&c)의 simulation>

시뮬레이션 코드를 run하여 시뮬레이션을 구현하였다. 위에서 조사한 진리표와 비교해본다면 동일한 결과값을 갖는다는 것을 확인할 수 있다.



<사진32. ~((a|b)&c)의 schematics>

~((a|b)&c) 을 schematic으로 나타내었다.

(~a&~b)|(~c)와 ~((a|b)&c)의 결과값을 비교하면 다음과 같다.

테이블이(가) 표시된 사진

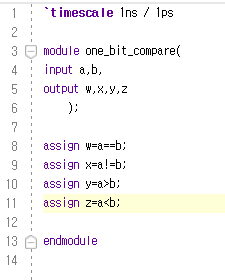
자동 생성된 설명

<표8. (~a&~b)|(~c)와 ~((a|b)&c)의 진리표>

위의 결과값을 비교한다면, 입력값 a,b,c에 따른 출력값이 같은 것을 알 수 있다. 따라서 (~a&~b)|(~c) = ~((a|b)&c)임을 알 수 있다.

4.

1비트 비교기를 베릴로그로 구현한 코드 및 결과값이다.



<사진33. 1비트 비교기의 design source>

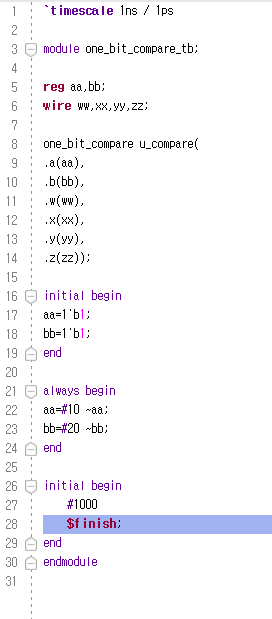
위 사진은 1비트 비교기에 대한 디자인 소스 파일이다. 입력값으로 a,b를 선언하고, 출력값으로 w,x,y,z를 선언하였다. 이후 출력값 w에 a==b, x에 a!=b, y에 a>b, z에 a<b을 각각 할당하였다.

테이블이(가) 표시된 사진

자동 생성된 설명

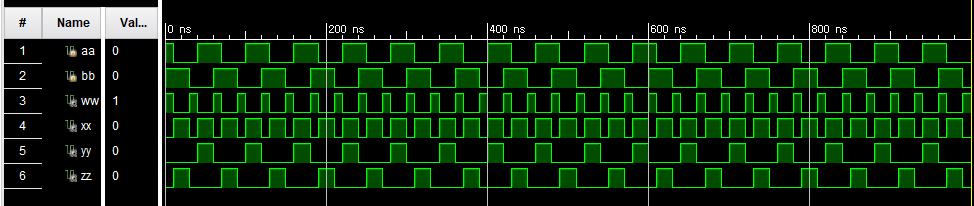
<표9. 1비트 비교기의 truth table>

1비트 비교기에 대한 결과값을 진리표로 나타내었다. A==b는 두 값이 같은 경우 1을 다를 경우 0을 반환하며, a!=b는 두 값이 다를 경우 1을 같을 경우 0을 반환한다. A>b는 a의 값이 b의 값보다 클 경우 1을 아닐 경우 0을 반환하며, a<b는 b의 값이 a의 값보다 클 경우 1을 아닐 경우 0을 반환한다.



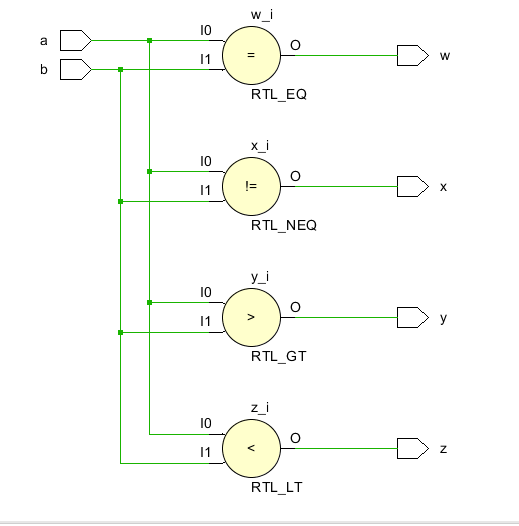
<사진34. 1비트 비교기의 simulation source>

1비트 비교기를 시뮬레이션으로 구현하기 위한 시뮬레이션 코드이다.



<사진35. 1비트 비교기의 simulation>

시뮬레이션 코드를 run하여 시뮬레이션을 구현하였다. 위에서 조사한 진리표와 비교해본다면 동일한 결과값을 갖는다는 것을 확인할 수 있다.



<사진36. 1비트 비교기의 schematics>

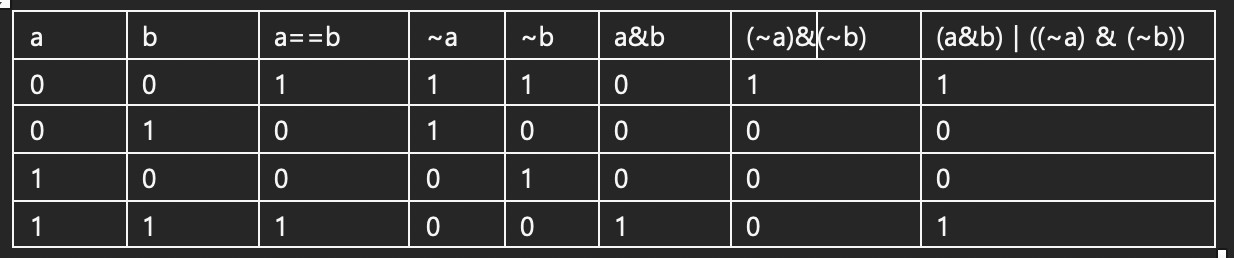
1비트 비교기를 schematic으로 나타내었다.

5.

드 모르간의 제1법칙과 제2법칙, Boolean function, 1비트 비교기를 실습을 통해 구현하였다. Design source file과 simulation source file을 구현하고 시뮬레이션과 진리표를 이용해 결과값을 확인하였다. 그 결과 드 모르간의 법칙은 (a&b)’=(a’|b’), (a|b)’=(a’&b’)를 만족함을 확인하였다. Boolean function에서는 (a’|b’)&c’과 ((a&b)|c)’이 같은 결과값을 가지고, (a’&b’)|c’과 ((a|b)&c)’이 같은 결과값을 가짐을 알 수 있었다. 1비트 비교기를 구현하여 베릴로그 언어에서의 ==, !=, <, > 연산자를 구현하였다. 결과가 참이라면 1을 거짓이라면 0을 반환하는 것은 다른 언어와 동일하다.

6.

1비트 비교기에서 사용한 연산자 ==, !=, <, >을 대수식으로 나타낼 수 있다.

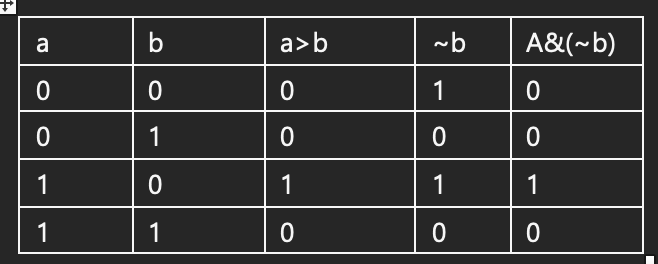
a==b를 (a&b) | ((~a) & (~b))와 같이 표현할 수 있다. 이를 진리표로 나타내면 다음과 같다.

A==b의 결과값과 (a&b) | ((~a) & (~b))의 결과값이 모든 경우의 수에 대해 동일하며, 결과값이 두 값이 같을 때 1을, 다를 때 0을 반환하고 있다.

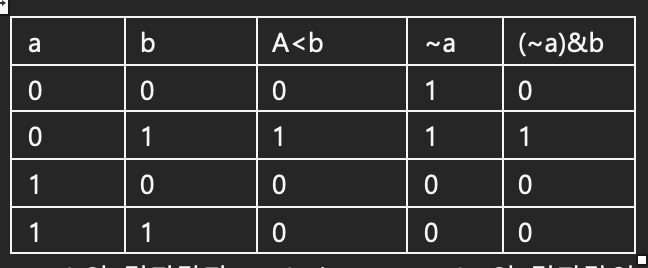
A!=b를 ((~a)&b) | (a & (~b))와 같이 표현할 수 있다. 이를 진리표로 나타내면 다음과 같다.텍스트, 점수판이(가) 표시된 사진

자동 생성된 설명

A!=b의 결과값과 ((~a)&b) | (a & (~b))의 결과값이 모든 경우의 수에 대해 동일하며, 결과값이 두 값이 다를 때 1을, 같을 때 0을 반환하고 있다.

A>b를 a&(~b)와 같이 표현할 수 있다. 이를 진리표로 나타내면 다음과 같다.

A>b의 결과값과 a&(~b)의 결과값이 모든 경우의 수에 대해 동일하며, A보다 b보다 큰 경우 1을, 나머지 경우에 0을 반환하고 있다.

A<b를 (~a)&b와 같이 표현할 수 있다. 이를 진리표로 나타내면 다음과 같다.

A==b의 결과값과 (a&b) | ((~a) & (~b))의 결과값이 모든 경우의 수에 대해 동일하며, 결과값이 두 값이 같을 때 1을, 다를 때 0을 반환하고 있다.