Lab1. 논리회로 기초

20200437 김채현

1. lab1_1. AND 게이트 구현

1) 개요

본 실험은 AND 게이트를 Verilog로 구현하는 것을 목표로 한다. inA, inB를 입력으로 하여 AND 게이트를 지나고 outAND를 출력하는 것을 구현했으며, 이를 Testbench를 작성한다. Testbench는 시뮬레이션을 수행하는데, 10ns 동안 Input A는 1ns마다, Input B는 2ns마다 한 번씩 반전시킨다. Input 둘 다 초기값은 1로 설정해주었으며 시뮬레이션을 통해 이를 시각적으로 확인하였다. 또한 모든 모듈은 Gate-Level Modeling으로 구현하였다.

2) 이론적 배경

논리게이트란 여러 개의 트랜지스터, 저항, 다이오트를 이용하여 특정 조건에서 특정한 출력을 내도록 전류의 흐름을 제어하도록 구현한 회로이다. 논리식을 실제 전자 회로로 구현할 때 참과 거짓을 High (높은 전압)와 Low (낮은 전압)에 대응할 수 있는데, Positive logic (Active high)는 high voltage를 1, low voltage를 0으로 생각하여 구현한 논리 회로이며, 반대로 negative logic은 low volatage가 0, high voltage가 1이 되도록 구현한 논리 회로이다.

AND 게이트는 논리곱을 구현하는 기본 디지털 논리 게이트로 입력 "X","Y"과 출력 "Z"인 AND 게이트는 Z=X·Y라는 논리식을 구현한다. 이를 기호와 truth table로 나타내면 아래와 같다.

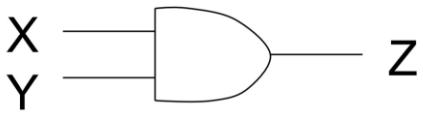


그림 1. AND gate 기호

X	Y	Z
0	0	0
0	1	0
1	0	0
1	1	1

그림 2. AND gate truth table

3) 실험 준비

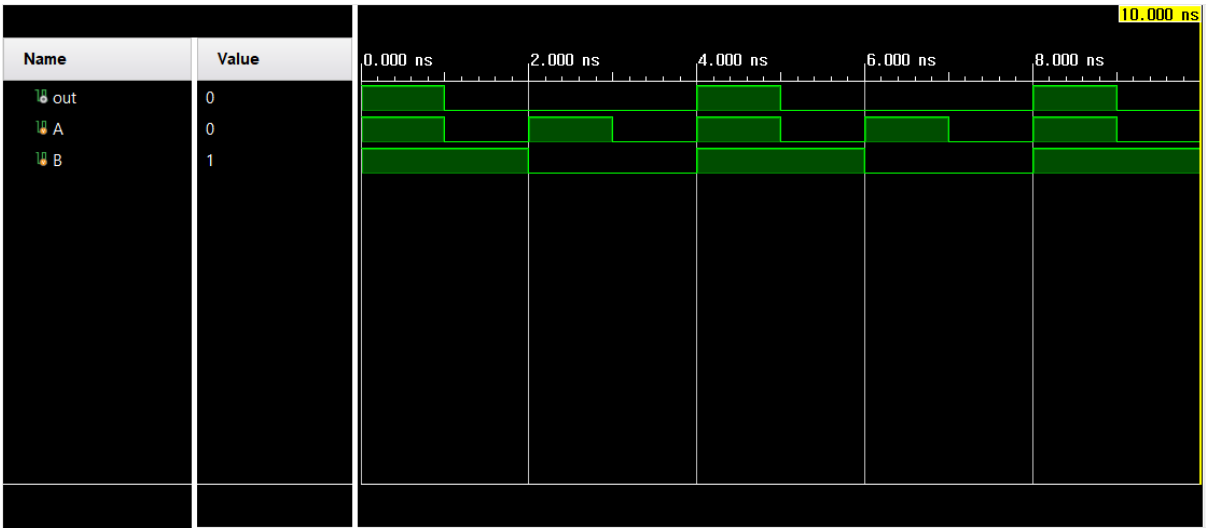
lab1_1.v에서는 module의 I/O ports에서 wire data type으로 output outAND와 input inA, inB를 선언해준 다음 Gate and를 이용하여 2 Input gate를 Gate-Level Modeling으로 구현하였다. Testbench code인 lab1_1tb.v에서는 입력 A, B와 출력 out인 lab1_1의 AND gate에 대해 시뮬레이션을 실행

하였다. 10ns 동안 시뮬레이션을 수행하며 두 입력인 A, B에 대해 A는 1ns마다, B는 2ns마다 한 번씩 반전시킨다. A, B 둘 다 0ns에서의 초기값은 1로 설정해주었다.

schematic은 생략하였다.

4) 결과

Testbench를 작성하여 이를 시뮬레이션 해주었다. 시각적으로 알 수 있듯 A는 1ns마다 값이 반전되고, B는 2ns마다 값이 반전된다. 따라서 AND gate의 출력인 out은 4ns에 한 번씩 1 값을 1ns 동안 가진다.



5) 논의

본 실험의 경우 결과를 simulation으로 잘 확인할 수 있었고, 결과 또한 만족스러웠다. 실험에서 어려웠던 점은 처음 Verilog를 사용하는 탓에 문법들이 익숙하지 않았던 것인데, lab을 하다 보니 익숙해진 것 같다.

2. lab1_2. Functionally complete 집합 구현

1) 개요

본 실험은 주어진 연산만을 이용하여 궁극적으로 {AND, OR, NOT} 집합을 완성하는 것을 목표로 한다. i에서는 OR, NOT gate만을 이용하여 AND gate를 구현했으며, ii에서는 반대로 AND, NOT gate만을 이용하여 OR gate를 구현한다. Iii과 iv에서는 각각 NAND, NOR gate를 이용하여 AND, OR, NOT를 구현한다. 마지막으로 이를 각각 schematic 기능을 이용하여 회로로 확인한다.

2) 이론적 배경

위에서 언급했던 AND 게이트는 논리곱을 구현하는 기본 디지털 논리 게이트로 입력 "X","Y"과 출력 "Z"인 AND 게이트는 Z=X·Y라는 논리식을 구현한다. OR 게이트는 논리곱을 구현하는 게이트로 입력 "X","Y"과 출력 "Z"인 OR 게이트는 Z=X+Y라는 논리식을 구현한다. NOT 게이트는 논리 회로 소자의 하나로, 출력이 입력과 반대되는 값을 가지는 논리소자이다. 입력 "X"가 NOT 게이트를 지나면 X'를 출력한다. NAND 게이트는 모든 입력이 참일 때에만 거짓인 출력을 내보내는 논리 회로이다. NOR 게이트는 주어진 입력이 모두 거짓인지 보는 논리 연산인 부정논리합을 구현하는 디지털 논리 회로이다. 다음은 각 게이트의 기호이다. AND 게이트의 기호는 lab1_1의 이론적 배경에서 언급했기에 생략한다.

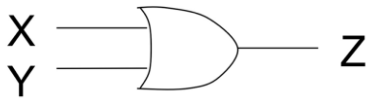


그림 3. OR gate 기호

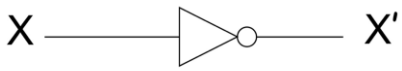


그림 4. NOT gate 기호

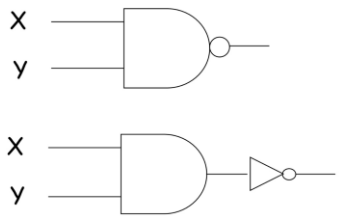


그림 5. NAND gate 기호

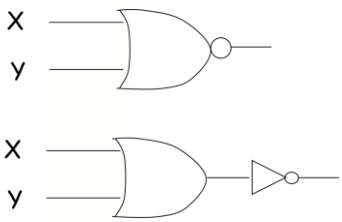


그림 6. NOR gate 기호

OR과 NOT을 이용하여 AND를 구현하기 위해서는 AB = (A' + B')'를 이용한다.

AND과 NOT을 이용하여 OR을 구현하기 위해서는 A+B=(A'B')'를 이용한다.

NAND를 이용하여 AND을 구현하기 위해서는 {(AB)'}'=AB를, OR을 구현하기 위해서는 (A'B')'=A+B를, NOT을 구현하기 위해서는 (AA)'=A'을 이용한다.

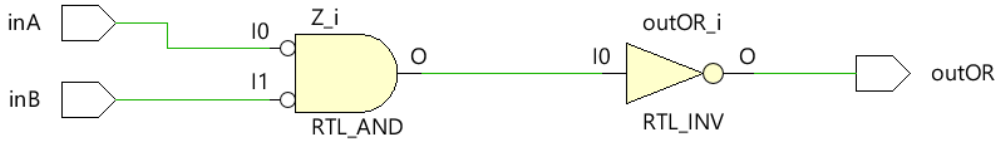
NOR을 이용하여 AND을 구현하기 위해서는 {(A+A)'+(B+B)'}'=AB를, OR을 구현하기 위해서는 {(A+B)'+(A+B)'}'=A+B를, NOT을 구현하기 위해서는 (A+A)'=A'을 이용한다.

3) 실험 준비

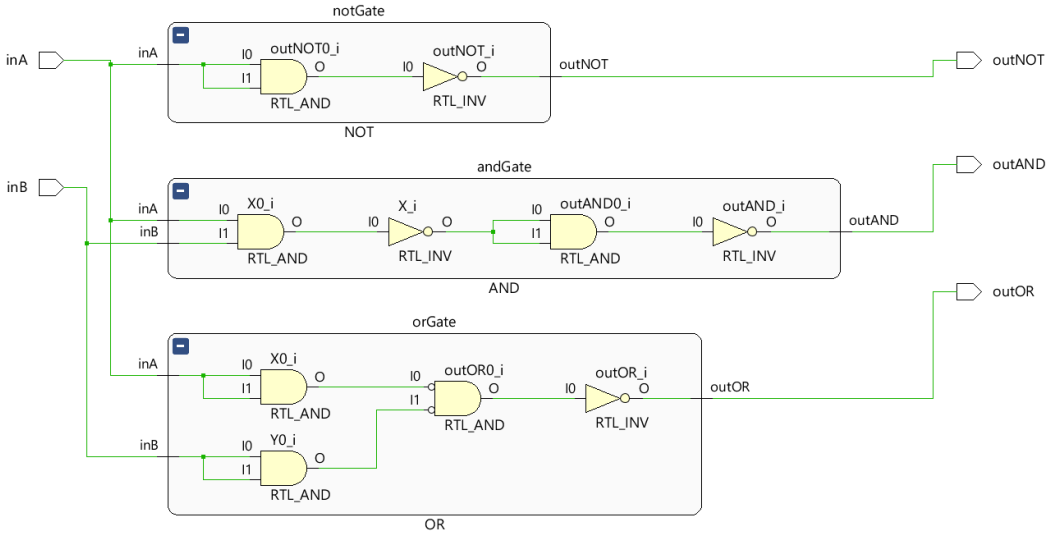
위의 이론적 배경을 이용하여 Gate-Level Modeling을 하고 RTL Analyzer Schematic을 수행한다.

아래는 i의 결과로 입력 inA, inB에 대하여 OR gate과 NOT gate을 이용하여 AND gate의 출력값과 동일한 출력 outAND를 구현하였다. 수식 AB = (A' + B')'을 이용하였다.

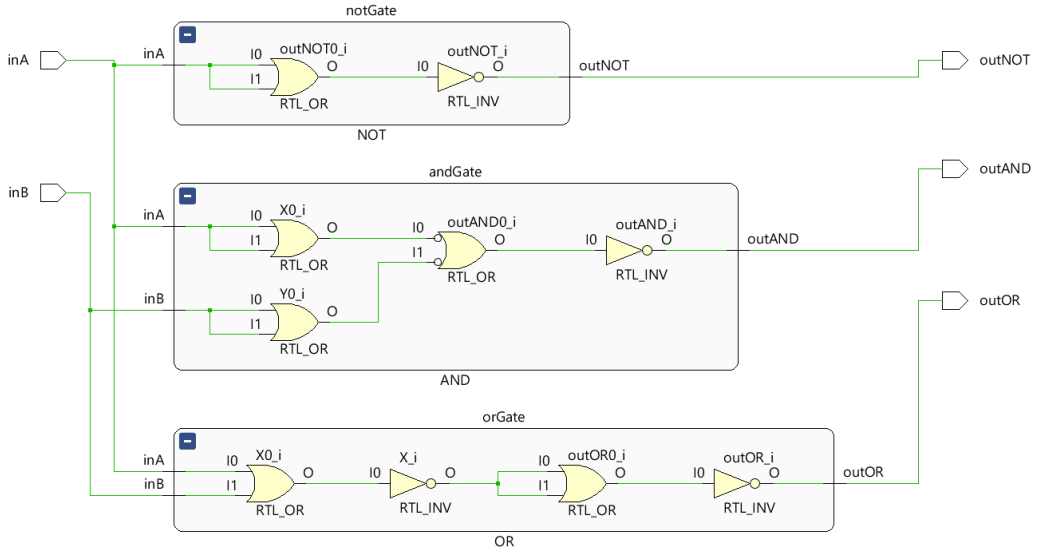
아래는 ii의 결과로 입력 inA, inB에 대하여 AND gate과 NOT gate을 이용하여 OR gate의 출력값과 동일한 출력 outOR을 구현하였다. 수식 A+B=(A'B')'을 이용하였다.



아래는 iii의 결과로 입력 inA, inB에 대하여 NAND gate만을 이용하여 각각 NOT gate, AND gate, OR gate의 출력값과 동일한 출력 outNOT, outAND, outOR을 구현하였다. 수식은 AND을 구현하기 위해서는 {(AB)'}'=AB를, OR을 구현하기 위해서는 (A'B')'=A+B를, NOT을 구현하기 위해서는 (AA)'=A'을 이용하였다.



아래는 iv의 결과로 입력 inA, inB에 대하여 NOR gate만을 이용하여 각각 NOT gate, AND gate, OR gate의 출력값과 동일한 출력 outNOT, outAND, outOR을 구현하였다. 수식은 AND을 구현하기 위해서는 {(A+A)'+(B+B)'}'=AB를, OR을 구현하기 위해서는 {(A+B)'+(A+B)'}'=A+B를, NOT을 구현하기 위해서는 (A+A)'=A'을 이용하였다.



4) 결과

실험 결과 값을 대입해보았을 때 각각 구현하고자 하는 gate의 값이 잘 출력되는 것을 확인할 수 있었다. Simulation의 경우 생략하였다.

5) 논의

본 실험의 경우 결과를 schematic을 이용하여 회로를 확인할 수 있었고, 결과 또한 만족스러웠다. 실험 도중 아직 vivado program이 익숙하지 않아 RTL Analyzer Schematic에 어려움을 겪었는데, sources를 추가할 때 design sources가 아닌 simulation sources로 하여 문제가 생겼었다. 또한 다른 source가 out-of-date가 되어 reload하지 않고 RTL Analyzer Schematic 실행하니 원하는 모듈이 아닌 다른 모듈의 schematic이 뜨는 문제도 발생하였다. 하지만 문제를 잘 해결하여, 원하는 회로의 결과를 시각적으로 잘 확인할 수 있었다.

3. 참고자료

위키백과 – AND 게이트, OR 게이트, NOT 게이트, NAND 게이트, NOR 게이트

그림 – 디지털시스템설계 chap2. Two-Level Combinational Logic