디지털시스템설계 Lab1 보고서

20230024 문요준

1) AND 게이트 구현 - lab1_1.v, lab1_1_tb.v

- lab1_1.v

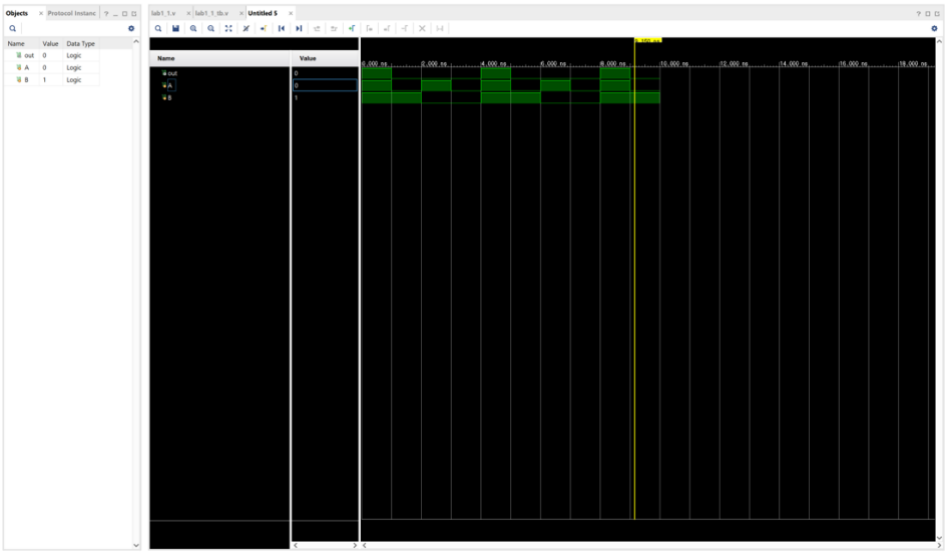
기본적인 AND 게이트를 구현하는 코드이다. 이 모듈은 두 입력 inA와 inB를 받아서 두 입력이 모두 1일 때만 출력 outAND가 1이 되는 기본적인 AND 게이트를 구현한다. 입력과 출력은 모두 와이어 타입으로 선언된다. assign 키워드를 사용하여 outAND 출력은 입력 inA와 inB의 논리적 AND 연산의 결과로 정의된다.

- lab1_1_tb.v

이 테스트 벤치는 lab1_1 모듈의 인스턴스를 생성하고, 입력 신호 A와 B에 다양한 값을 적용하여 출력 out을 관찰한다.

- 초기화: 테스트 벤치의 initial 블록에서는 먼저 A와 B를 1로 설정한다. 이는 AND 게이트의 동작을 테스트하기 위한 초기 조건이다. 이후, 10나노초 후에 시뮬레이션을 종료하는 $finish 명령이 있다.
- A의 플립: always 블록을 사용하여 A 신호를 매 1나노초마다 반전시킨다. 이를 통해 A의 모든 가능한 값을 테스트한다.
- B의 플립: 또 다른 always 블록에서는 B 신호를 매 2나노초마다 반전시킨다. 이는 B에 대해서도 모든 가능한 값 변화를 테스트하게 해준다.

- lab1_1_tb.v 실행 결과

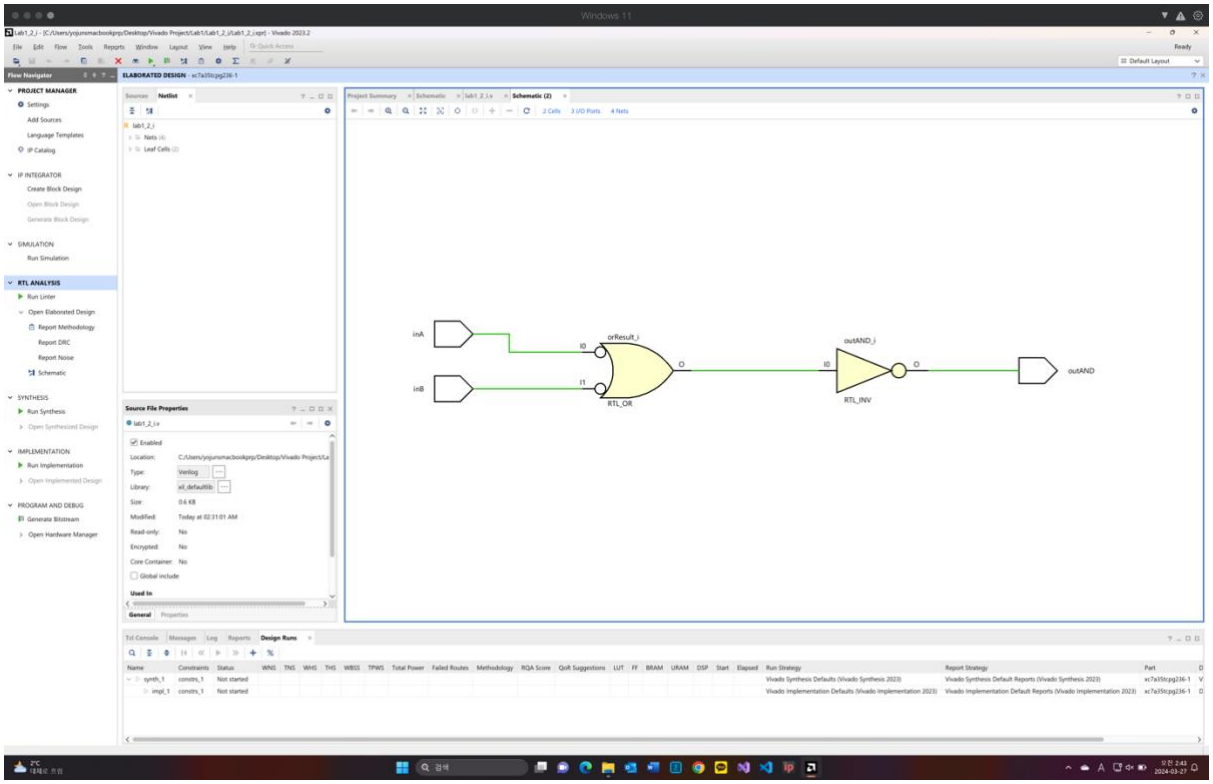


2) Functionally complete 집합 구현 - lab1_2_i.v, lab1_2_ii.v, lab1_2_iii.v, lab1_2_iv.v

- lab1_2_i.v

- 입력과 출력: 모듈 lab1_2_i는 두 입력 inA와 inB를 받아서 한 개의 출력 outAND를 제공한다. 입력과 출력은 모두 와이어 타입으로 선언된다.
- 내부 와이어 선언: 내부적으로, 이 구현은 중간 결과를 전달하기 위해 세 개의 추가 와이어 notA, notB, 그리고 orResult를 사용한다.
- NOT 게이트 구현: 입력 inA와 inB 각각에 대한 NOT 연산을 수행하여 notA와 notB를 생성한다. 이는 입력된 두 값의 반대값을 생성한다.
- OR 게이트 구현: notA와 notB를 입력으로 받는 OR 게이트를 구현하여 orResult를 생성한다. 이 단계는 드모르간의 법칙에 따라 NOT A OR NOT B 연산을 수행하는 것과 같다.
- 최종 NOT 게이트를 통한 AND 결과 생성: orResult에 NOT 연산을 한 번 더 적용하여 최종적으로 outAND 출력을 생성한다. 이는 드모르간의 법칙을 이용해 (NOT A OR NOT B)의 반대, 즉 A AND B 연산 결과를 얻는 과정이다.

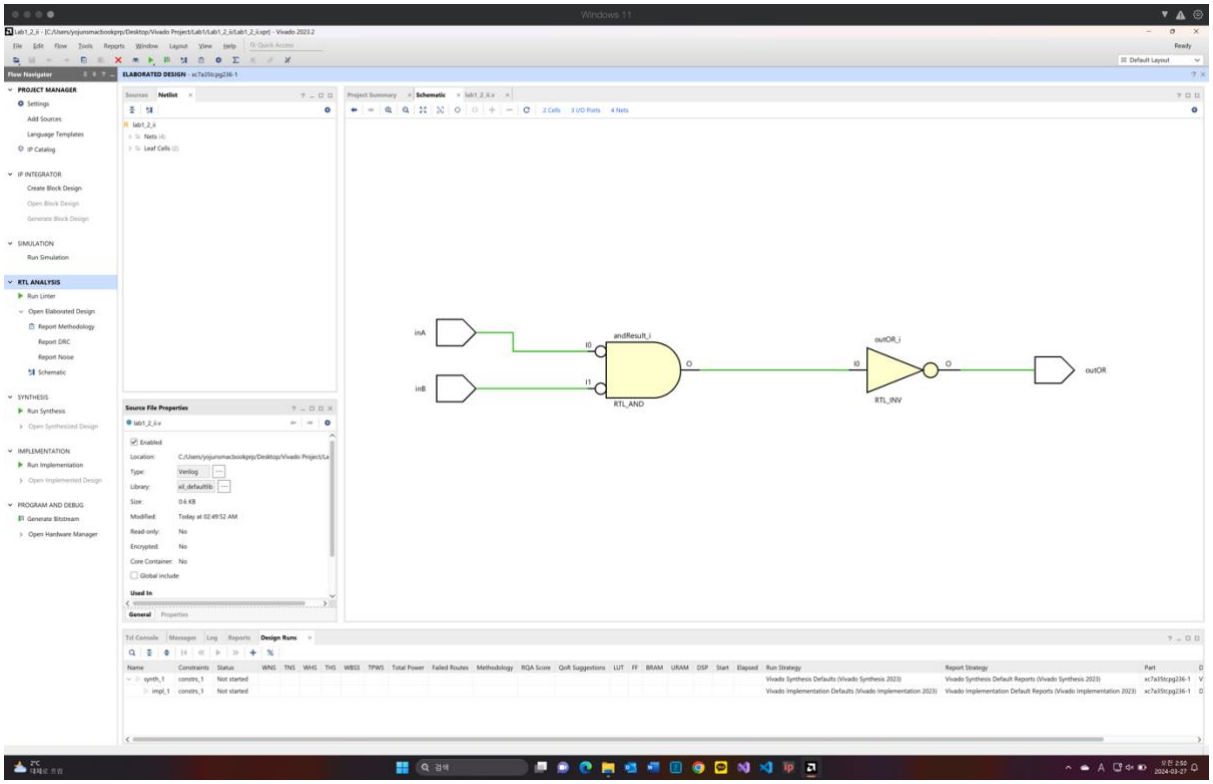
- lab1_2_i.v 실행 결과



- lab1_2_ii.v

- 입력과 출력: lab1_2_ii 모듈은 두 개의 입력 신호 inA와 inB를 받고, 하나의 출력 신호 outOR를 가진다. 모든 신호는 와이어 타입으로 선언된다.
- 내부 와이어 선언: 이 구현에서는 중간 계산을 위해 세 개의 추가 와이어 notA, notB, 그리고 andResult를 사용한다.
- NOT 게이트 구현: 입력 inA와 inB에 대해 각각 NOT 연산을 수행하여 notA와 notB를 생성한다. 이 과정은 두 입력의 반대값을 얻는다.
- AND 게이트 구현: notA와 notB를 AND 게이트의 입력으로 사용하여 andResult를 생성한다. 이 단계는 드모르간의 법칙에 따라 NOT A AND NOT B 연산을 수행하는 것과 같다.
- 최종 NOT 게이트로 OR 결과 생성: andResult에 NOT 연산을 적용하여 최종 출력인 outOR를 생성한다. 이는 드모르간의 법칙을 이용해 NOT(NOT A AND NOT B)를 계산하는 것이며, 이는 A OR B의 논리적 동등함을 의미한다.

- lab1_2_ii.v 실행 결과



- **lab1_2_iii.v**

- 입출력 정의: 이 모듈은 두 개의 입력 inA와 inB, 그리고 세 개의 출력 outAND, outOR, outNOT을 가진다.
- 하위 모듈 호출: AND, OR, NOT 게이트 각각의 기능을 구현하는 하위 모듈(AND, OR, NOT)을 인스턴스화한다. 각 게이트는 입력에 따라 해당 논리 연산의 결과를 출력으로 제공한다.

AND 모듈

- NAND 게이트를 이용한 AND 구현: 먼저 두 입력 inA와 inB에 대한 NAND 연산을 수행한다. NAND 게이트의 출력을 다시 같은 NAND 게이트의 입력으로 연결하면 AND 연산의 결과를 얻을 수 있다. 이는 NAND 게이트가 유니버설 게이트로서 다른 모든 논리 게이트를 구현할 수 있음을 보여준다.

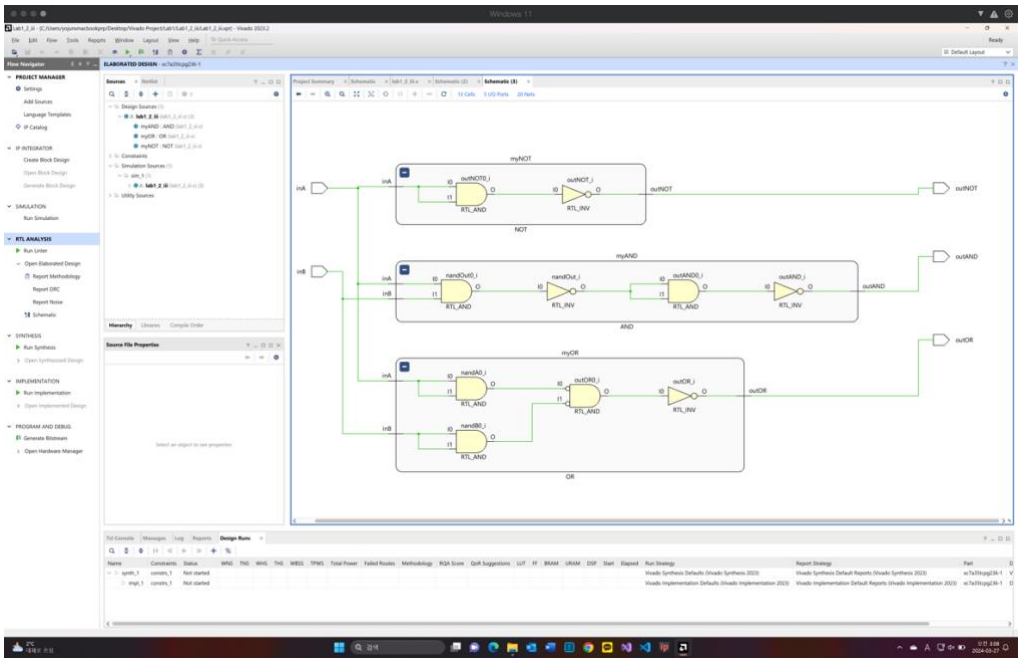
OR 모듈

- NAND 게이트를 이용한 OR 구현: 각 입력에 대해 자기 자신과의 NAND 연산을 수행하여 NOT 연산을 구현한다(nandA는 NOT A, nandB는 NOT B). 이후 nandA와 nandB에 대한 NAND 연산을 수행하여 OR 연산의 결과를 얻는다. 이는 드모르간의 법칙을 이용한 것과 유사하다.

NOT 모듈

- NAND 게이트를 이용한 NOT 구현: 단일 입력에 대해 자기 자신과의 NAND 연산을 수행하면 NOT 연산의 결과를 얻을 수 있다. 이는 입력의 반대값을 출력으로 제공한다.

- **lab1_2_iii.v** 실행 결과



- lab1_2_iv.v

- 입출력 정의: 이 모듈은 두 개의 입력 inA와 inB와 세 개의 출력 outAND, outOR, outNOT을 가진다.
- 하위 모듈 호출: AND, OR, NOT 각각의 기능을 구현하는 하위 모듈을 인스턴스화한다. 이 모듈들은 각각의 논리 연산 결과를 출력으로 제공한다.

AND 모듈

- NOR 게이트를 이용한 AND 구현: 입력 inA와 inB에 대해 각각 NOT 연산을 수행한다(이를 위해 입력과 자신을 NOR 연산). 그 다음, 이 두 NOT 결과에 대해 NOR 연산을 수행하고, 그 결과를 다시 NOR 연산하여 AND 연산의 결과를 얻는다.

OR 모듈

- NOR 게이트를 이용한 OR 구현: 입력 inA와 inB에 대해 각각 NOT 연산을 수행한다(이 역시 NOR를 사용). 이 두 NOT 결과에 대해 NOR 연산을 수행하여 OR 연산의 결과를 얻는다.

NOT 모듈

- NOR 게이트를 이용한 NOT 구현: 단일 입력에 대해 자기 자신과의 NOR 연산을 수행하면 NOT 연산의 결과를 얻을 수 있다. 이 방법은 입력의 반대값을 출력으로 제공한다.

- lab1_2_iv.v 실행 결과