4주차 결과보고서

전공: 컴퓨터공학과 학년: 3학년 학번: 20211547 이름: 신지원

**1.**

NAND gate, NOR gate, XOR gate, AOI gate의 동작을 이해하고, Verilog를 사용하여 4개의 input, 3

개의 output에 대한 NAND gate, NOR gate, XOR gate, AOI gate를 구현하는 것을 목적으로 하였다. 구현한 코드를 바탕으로 simulation을 시각화하고 나아가 schema 까지 살펴보는 것을 목적으로 하였다.

**2.**

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

<4-input NAND gate 의 inv, inv\_tv 코드>

inv 코드에선, input 4개를 a, b, c, d 로 잡고 그에 따른 output 3개를 x, y, z 로 선언하였다. NAND gate 는 AND에 NOT을 붙인 것과 같기 때문에 ‘ ~( input1 & input2 )’ 으로 구현하였다. 참고자료에서는 a와 b의 NAND gate 출력을 다시 C와 NAND gate 로 출력하고 이를 다시 d와 NAND gate 출력하였다. inv\_tb 코드에서는 입력의 주기를 선언해주었는데, 모든 출력을 보고자 하여 #20, #30, #40, #50 으로 주기를 달리하였다.

스크린샷, 디스플레이, 소프트웨어, 텍스트이(가) 표시된 사진

자동 생성된 설명

<4-input NAND gate 의 simulation >

x는 aa와 bb 를 교집합에 부정이기 때문에 둘 다 1인 경우에 0, 둘 중 하나라도 1이 아닌 경우에 1을 출력할 것이다. y는 x와 cc의 교집합에 부정이기 때문에 x와 마찬가지로 출력할 것이다. aa, bb 까지 고려하자면 aa 와 bb 중 하나라도 0이면서 cc가 1일 때 y는 0을 출력할 것이다. cc가 0이거나 aa와 bb가 모두 1이라면 1을 출력할 것이다. 마지막 출력값인 z를 통해 4-input NAND gate 결과를 알 수 있는데, z는 y와 dd의 교집합의 부정이기 때문에, y의 조건에서 dd 의 조건까지 추가되었다고 생각할 수 있다.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **In a** | **In b** | **In c** | **In d** | **out x** | **out y** | **out z** |
| 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| 0 | 0 | 0 | 1 | 1 | 1 | 0 |
| 0 | 0 | 1 | 0 | 1 | 1 | 1 |
| 0 | 0 | 1 | 1 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 | 0 | 1 | 0 |

**3.**

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

<4-input NOR gate 의 inv, inv\_tv 코드>

inv 코드에선, input 4개를 a, b, c, d 로 잡고 그에 따른 output 3개를 x, y, z 로 선언하였다. NOR gate 는 OR에 NOT을 붙인 것과 같기 때문에 ‘ ~( input1 | input2 )’ 으로 구현하였다. 참고자료에서는 a와 b의 NOR gate 출력을 다시 C와 NOR gate 로 출력하고 이를 다시 d와 NOR gate 출력하였다. inv\_tb 코드에서는 입력의 주기를 선언해주었는데, 모든 출력을 보고자 하여 #20, #30, #40, #50 으로 주기를 달리하였다.

스크린샷, 소프트웨어, 멀티미디어 소프트웨어, 텍스트이(가) 표시된 사진

자동 생성된 설명

<4-input NOR gate 의 simulation >

x는 aa와 bb 를 합집합에 부정이기 때문에 둘 다 0인 경우에 1, 둘 중 하나라도 0이 아닌 경우에 0을 출력할 것이다. y는 x와 cc의 합집합에 부정이기 때문에 x와 마찬가지로 출력할 것이다. aa, bb 까지 고려하자면 aa 와 bb 가 둘 다 0 이거나 cc가 1일 때 y는 0을 출력할 것이다. 따라서 aa와 bb가 하나라도 1이면서 cc가 0이라면 1을 출력할 것이다. 마지막 출력값인 z를 통해 4-input NOR gate 결과를 알 수 있는데, z도 마찬가지로 y와 dd의 합집합의 부정이기 때문에, y의 조건에서 dd 의 조건까지 추가되었다고 생각할 수 있다.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **In a** | **In b** | **In c** | **In d** | **out x** | **out y** | **out z** |
| 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 |

**4.**

텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

자동 생성된 설명텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

<4-input XOR gate 의 inv, inv\_tv 코드>

inv 코드에선, input 4개를 a, b, c, d 로 잡고 그에 따른 output 3개를 x, y, z 로 선언하였다. XOR gate 는 ^ 기호를 사용하기 때문에 ‘ input1 ^ input2 ’ 으로 구현하였다. 참고자료에서는 a와 b의 XOR gate 출력을 다시 C와 XOR gate 로 출력하고 이를 다시 d와 XOR gate 출력하였다. inv\_tb 코드에서는 입력의 주기를 선언해주었는데, 모든 출력을 보고자 하여 #20, #30, #40, #50 으로 주기를 달리하였다.

스크린샷, 멀티미디어 소프트웨어, 소프트웨어, 그래픽 소프트웨어이(가) 표시된 사진

자동 생성된 설명

<4-input XOR gate 의 simulation >

XOR gate 는 1이 홀수 개일 때 1을 출력하는 논리회로다. 따라서 x 는 aa 와 bb 중 입력이 하나만 1일 때, y는 aa, bb, cc 중 입력이 하나 또는 세 개가 모두 1일 때, z는 aa, bb, cc, dd 중 입력이 하나 또는 세 개가 1일 때 출력이 1일 것이다. 시뮬레이션으로 보았을 때, 처음 20초부터 bb에 신호가 들어오기 전까지 x, y, z 모두 1을 출력한다. 따라서 위와 같이 주기를 정해준 시뮬레이션에서는 z가 켜질 때는 x도 켜질 수 밖에 없다.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **In a** | **In b** | **In c** | **In d** | **out x** | **out y** | **out z** |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 | 0 | 1 | 0 |

**5.**

텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

자동 생성된 설명텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

<4-input AOi gate 의 inv, inv\_tv 코드>

inv 코드에선, input 4개를 a, b, c, d 로 잡고 그에 따른 output 3개를 x, y, z 로 선언하였다. AOi gate 는 and 와 or 그리고 not gate 를 사용하기 때문에 ‘ ~ [( input1 & input2 ) | ( input3 & input 4)]’ 으로 구현하였다. 참고자료에서는 a와 b의 AND gate 출력과 C와 D의 AND gate 출력을 OR gate 로 출력하여 inverter 해주었다. inv\_tb 코드에서는 입력의 주기를 선언해주었는데, 모든 출력을 보고자 하여 #20, #30, #40, #50 으로 주기를 달리하였다.

스크린샷, 소프트웨어, 디스플레이, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

<4-input AOi gate 의 simulation >

AND gate 를 통하여 각각 aa와 bb, cc와 dd 를 먼저 처리해주고 그 값들을 z 로 묶어주는 모습을 볼 수 있다. 따라서 시뮬레이션에서는 x 와 y 의 NOR gate 꼴이 z 에서 나타남을 볼 수 있다.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **In a** | **In b** | **In c** | **In d** | **out x** | **out y** | **out z** |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 0 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 0 |

6.

텍스트, 도표, 라인, 번호이(가) 표시된 사진

자동 생성된 설명

위는 4-input NAND gate , NOR gate, XOR gate 의 schema 이미지다. 위를 통해 a와 b의 출력값을 c 와, 그 값을 d 와 계산된 다는 점을 알 수 있다.

텍스트, 스크린샷, 도표, 번호이(가) 표시된 사진

자동 생성된 설명

위는 4-input AOI gate , NOR gate, XOR gate 의 schema 이미지다. 위를 통해 가장 먼저 보았단 schema 그림과는 다르게 a와 b, c와 d 가 먼저 계산되고 그 뒤에 두 값이 함께 계산됨을 알 수 있다.

뿐만 아니라 AND 에 NOT 을 붙인 값이 절차를 나누어 작성하지 않아도 ~(a & b) 로 간편하게 작성할 수 있었다. 여러 논리회로를 붙여가며 새로운 논리회로를 만들 수 있다는 점이 흥미로웠다.

**7.**

Universal gate 는 한 종류의 gate 만으로도 모든 Boolean expression 을 표현할 수 있는 gate 를 말한다. (2input 일 때) 우리가 실습한 NAND gate, NOR gate 가 universal gate 의 예시다.