4주차 결과보고서

전공: 컴퓨터공학과 학년: 4학년 학번: 20191583 이름: 김태곤

**1. 실험 목적**

본 실험은 Vivado 프로그램과 Verilog HDL을 활용하여 기본 디지털 논리 게이트인 NAND, NOR, XOR과 AOI 설계 및 시뮬레이션을 통해 디지털 논리 설계의 기본 개념을 심화하고, 실제 FPGA를 통해 구현을 해보면서 이론과 실습의 연결고리를 강화하고 경험하는 것을 목적으로 한다. 이 과정에서, Verilog HDL을 사용하여 4-입력 NAND, NOR, XOR 게이트와 AOI를 직접 설계하고, 이를 Vivado 프로그램을 통해 시뮬레이션함으로써 각 논리 게이트의 동작 원리와 논리식에 대한 깊은 이해를 목표로 한다. 또한, 다양한 입력 조합에 대한 시뮬레이션을 통해 구현된 논리 회로의 동작을 확인하고, 이를 통해 설계한 회로의 정확성을 평가한다.

**2. 4-input NAND gate의 simulation 결과 및 과정에 대해서 설명하시오.**

**(4 input, 3 output)[4장 ppt 6,7 page 참조 , 진리표 작성]**

**텍스트, 스크린샷, 소프트웨어, 번호이(가) 표시된 사진

자동 생성된 설명1) Design Source와 Testbench file**

**텍스트, 스크린샷, 디스플레이, 소프트웨어이(가) 표시된 사진

자동 생성된 설명**

**2) Simulation 결과**

전자제품, 스크린샷, 컴퓨터, 텍스트이(가) 표시된 사진

자동 생성된 설명

**3) 진리표**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| A | B | C | D | E | F | G |
| 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| 0 | 0 | 0 | 1 | 1 | 1 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 | 1 | 0 |
| 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 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 | 0 | 1 |
| 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 |

**4) 구현 과정**

입력 A, B, C, D와 출력 E, F, G가 주어졌을 때, A와 B의 NAND 연산 결과가 출력 E에 할당되고, 이후 C와 E의 NAND 연산 결과가 출력 F에, 마지막으로 D와 F의 NAND 연산 결과가 출력 G에 할당된다. 이 구조에서는 각 단계별로 두 입력 값이 모두 1일 경우에만 0이 출력되고, 그 외의 경우에는 1이 출력된다. 이와 같은 특성 때문에, 4input, 3output NAND 게이트의 최종 출력 G는 4input, 1output NAND 게이트와 다른 결과 패턴을 보인다. 4input, 1output NAND 게이트는 모든 입력 값이 1인 경우 0을 출력하고, 그 외에는 1을 출력한다. 4input, 3output은 각 단계별로 NAND를 거치므로 위의 진리표와 같은 결과가 나온다. 해당 simulation 결과는 2-1의 Design Source와 Testbench의 실행을 통해 얻은 결과이다.

**3. 4-input NOR gate의 simulation 결과 및 과정에 대해서 설명하시오.**

**(4 input, 3 output)[4장 ppt 8,9 page 참조 , 진리표 작성]**

**텍스트, 스크린샷, 번호, 소프트웨어이(가) 표시된 사진

자동 생성된 설명 1) Design Source와 Testbench file**

**텍스트, 스크린샷, 디스플레이, 소프트웨어이(가) 표시된 사진

자동 생성된 설명**

**2) Simulation 결과**

스크린샷, 텍스트, 디스플레이, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

**3) 진리표**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| A | B | C | D | E | F | G |
| 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) 구현 과정**

입력 A, B, C, D와 출력 E, F, G가 주어졌을 때, A와 B의 NOR 연산 결과가 출력 E에 할당되고, 이후 C와 E의 NOR 연산 결과가 출력 F에, 마지막으로 D와 F의 NOR 연산 결과가 출력 G에 할당된다. 이 구조에서는 각 단계별로 모든 입력 값이 0일 경우에만 1이 출력되고, 그 외의 경우에는 0이 출력된다. 이와 같은 특성 때문에, 4input, 3output NOR 게이트의 최종 출력 G는 4input, 1output NOR 게이트와 다른 결과 패턴을 보인다. 4input, 1-output NOR 게이트는 모든 입력 값이 0인 경우에만 1을 출력하고, 그 외에는 0을 출력한다. 4input, 3output은 각 단계별로 NOR을 거치므로 위의 진리표와 같은 결과가 나온다. 해당 simulation 결과는 3-1의 Design Source와 Testbench의 실행을 통해 얻은 결과이다.

**4. 4-input XOR gate의 simulation 결과 및 과정에 대해서 설명하시오.**

**(4 input, 3 output)[3장 ppt 10,11 page 참조 , 진리표 작성]**

**텍스트, 스크린샷, 번호, 소프트웨어이(가) 표시된 사진

자동 생성된 설명 1) Design Source와 Testbench file**

**텍스트, 스크린샷, 디스플레이, 소프트웨어이(가) 표시된 사진

자동 생성된 설명**

**2) Simulation 결과**

스크린샷, 텍스트, 디스플레이, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

**3) 진리표**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| A | B | C | D | E | F | G |
| 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 |

**4) 구현 결과**

입력 A, B, C, D와 출력 E, F, G가 주어졌을 때, A와 B의 XOR 연산 결과가 출력 E에 할당되고, 이후 C와 E의 XOR 연산 결과가 출력 F에, 마지막으로 D와 F의 XOR 연산 결과가 출력 G에 할당된다. 이 구조에서는 각 단계별로 입력 값들 중 1의 입력이 홀수개인 경우에만 1이 출력되고, 그 외의 경우에는 0이 출력된다. 즉 입력된 값이 다른 경우 1을 반환한다. 기존에 살펴본 NAND와 NOR과는 다르게 XOR은 4input, 1output과 4input, 3output의 결과는 같다. G의 경우 입력된 1의 수가 홀수개인 경우에만 1의 결과값을 갖는다. 이를 통해 여러 단계에 걸쳐 XOR 연산을 수행하더라도, 전체 입력 중 홀수 개의 1이 있을 경우 최종 출력은 1이 되고, 그렇지 않은 경우 0이 되는 것을 알 수 있다. 해당 시뮬레이션 결과는 4-1의 Design Source와 Testbench의 실행을 통해 얻은 결과이다.

**5. 4-input AOI gate의 simulation 결과 및 과정에 대해서 설명하시오.**

**(4 input, 3 output)[4장 ppt 12,13 page 참조, 진리표 작성]**

**1) Design Source와 Testbench file**

**텍스트, 스크린샷, 디스플레이, 소프트웨어이(가) 표시된 사진

자동 생성된 설명텍스트, 스크린샷, 디스플레이, 소프트웨어이(가) 표시된 사진

자동 생성된 설명**

**2) Simulation 결과**

스크린샷, 텍스트, 컴퓨터, 디스플레이이(가) 표시된 사진

자동 생성된 설명

**3) 진리표**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| A | B | C | D | E | F | G |
| 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 |

**4) 구현 결과**

입력 A, B, C, D와 출력 E, F, G가 주어졌을 때, A와 B의 AND 연산 결과가 출력 E에 할당되고, 이후 C와 D의 AND 연산 결과가 출력 F에, 마지막으로 E와 F의 NOR 연산 결과가 출력 G에 할당된다. 이러한 구조로 인해 E와 F가 모두 0인 경우 1의 결과값을 갖는다. 즉 A와B쌍, C와D쌍 중 하나의 쌍이라도 동시에 둘다 1의 입력을 받게 되면 0의 결과를 갖게 된다. 해당 시뮬레이션 결과는 5-1의 Design Source와 Testbench의 실행을 통해 얻은 결과이다. 이러한 AOI(AND-OR-Invert) 구조의 구현 과정은 디지털 회로 설계에서 흔히 볼 수 있는 AND, OR, NOT 논리 게이트의 조합으로 이루어져 있으며, 특히 이 구조는 논리 게이트를 효율적으로 사용하여 복잡한 논리 함수를 구현하는 데 적합하다.

**6. 결과 검토 및 논의사항.**

3주차 실험의 AND, OR의 경우 1output과 3output의 결과는 같았다. 때문에 4주차 실험에서 NAND, NOR, XOR 또한 1output과 3output의 결과가 같을 것이라고 추측 했었다. 하지만 실험 결과 NAND와 NOR의 경우 다른 결과가 나왔고, XOR의 경우에만 같은 결과가 나왔다. 즉 NAND와 NOR은 결합법칙이 성립되지 않고, AND, OR, XOR은 결합법칙이 성립한다는 것을 알 수 있었다. 따라서 실제 회로 설계를 진행할 때 이 점을 유의해서 설계를 진행해야 하며, 이러한 성질들은 회로 설계에서 유용하게 사용될 것 같다. 또한 XOR 게이트의 경우 입력 받은 1의 수가 짝수인지, 홀수인지 쉽게 확인할 수 있으므로 유용하게 쓰일 수 있을 것 같다.

AOI 구조 또한 NAND나 NOR 게이트만을 사용하는 경우에 비해 게이트 수를 줄이면서도 동일한 논리 기능을 수행할 수 있어, 회로의 복잡성과 전력 소모를 줄일 수 있을 것 같다.

**7. 추가 이론 조사 및 작성.**

지금까지 우리는 AND, OR, NOT, NAND, NOR, XOR, AOI의 게이트 구조를 자세히 실습을 통해 알아보았다. 이 외에도 실습을 진행하지 않은 XNOR 게이트가 존재한다.

XNOR (Exclusive NOR) 게이트는 디지털 논리 회로에서 주로 사용되는 기본 논리 게이트 중 하나이다. XNOR 게이트는 두 입력이 서로 같을 때 1을 출력하고, 그렇지 않으면 0을 출력한다. 즉, 두 입력이 모두 1이거나 모두 0일 때만 1을 출력하는 특성을 가진다. 이는 XOR 게이트의 출력에 NOT 연산을 적용한 것과 같으며, XOR 게이트의 반대 동작을 수행한다. 아래와 같이 표현할 수 있으며, 실제 구현할 때는 아래 이미지처럼 구현한다.

스케치, 도표, 라인, 그림이(가) 표시된 사진

자동 생성된 설명도표, 라인, 평면도, 기술 도면이(가) 표시된 사진

자동 생성된 설명