4주차 결과보고서

전공: 컴퓨터공학과/생명과학과 학년: 9학기 학번: 20181435 이름: 박다희s

**1.**

이번 실험의 목적은 NAND, NOR, XOR, AOI(AND OR inverter) gate의 동작을 이해하고 이를 구현하는 Verilog 코드를 작성함으로 4종류의 gate가 어떻게 동작하는 지 그 원리를 이해하는 것이다. 그러기 위해서 Vivado 프로그램을 통해 Verilog언어로 논리식을 사용하여 NAND, NOR, XOR 그리고 AOI gate를 코드로 구현하고, simulation을 통해 우리가 구현한 코드가 여러 input값에 따라 어떻게 동작되는지 확인한다. Simulation으로 확인한 후에는 FPGA로 회로의 동작을 확인한다.

**2.**

**텍스트, 스크린샷, 영수증, 폰트이(가) 표시된 사진

자동 생성된 설명텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명**

**스크린샷, 사각형, 직사각형, 텍스트이(가) 표시된 사진

자동 생성된 설명**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Input a | Input b | Input c | Input d | Output e | Output f | Output 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 |

왼쪽 위에서부터 차례대로 testbench file, design source file, simulation 그리고 진리표이다. Source file을 보면 input a와 b의 NAND 값을 e에, e와 c의 NAND값을 f에, 그리고 f와 d의 NAND값을 g에 할당했고, g가 최종 출력물이 된다. NAND gate같은 경우는 들어온 입력 값이 모두 1이어야지만 출력 값으로 0을 내보내고, 입력 값 중에서 0이 한개라도 있다면 출력 값으로 1을 내보낸다. 따라서 이번에 구현한 회로는 최종 출력 값으로 0이 나오려면 순서상 마지막으로 들어가는 input인 d는 무조건 1이어야 한다. 그리고 input d와 NAND연산할 output f도 1이어야 하므로 (1, 1, 1, 1), (1, 1, 0, 1), (1, 0, 0, 1), (0, 1, 0, 1), (0, 0, 0, 1)의 경우에서만 최종 출력 값이 0이 나온다. 이는 simulation의 결과로도 확인할 수 있다. 코드로 구현할 때에는 design source file 사진을 보면 알 수 있듯이 입력 값 2개씩 묶어서 AND연산을 해준 뒤에 NOT을 붙였다. 이 과정을 3번 진행하여 4input 3output NAND연산자를 구현하였다.

**3.**

**텍스트, 영수증, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명**

**스크린샷, 도표, 사각형, 텍스트이(가) 표시된 사진

자동 생성된 설명**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Input a | Input b | Input c | Input d | Output e | Output f | Output 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 |

왼쪽 위에서부터 차례대로 testbench file, design source file, simulation 그리고 진리표이다. Source file을 보면 input a와 b의 NOR 값을 e에, e와 c의 NOR값을 f에, 그리고 f와 d의 NOR값을 g에 할당했고, g가 최종 출력물이 된다. NOR gate같은 경우는 들어온 입력 값이 모두 0이어야지만 출력 값으로 1을 내보내고, 입력 값 중에서 1이 한개라도 있다면 출력 값으로 0을 내보낸다. 따라서 이번에 구현한 회로는 최종 출력 값으로 1이 나오려면 순서상 마지막으로 들어가는 input인 d는 무조건 0이어야 한다. 그리고 input d와 NAND연산할 output f도 0이어야 하므로 (1, 1, 1, 0), (1, 0, 1, 0), (0, 1, 1, 0), (0, 0, 1, 0), (0, 0, 0, 0)의 경우에서만 최종 출력 값이 1이 나온다. 이는 simulation의 결과로도 확인할 수 있다. 코드로 구현할 때에는 design source file 사진을 보면 알 수 있듯이 입력 값 2개씩 묶어서 OR연산을 해준 뒤에 NOT을 붙였다. 이 과정을 3번 진행하여 4input 3output NOR연산자를 구현하였다.

**4.**

**텍스트, 스크린샷, 폰트, 영수증이(가) 표시된 사진

자동 생성된 설명텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명**

**스크린샷, 텍스트, 사각형, 도표이(가) 표시된 사진

자동 생성된 설명**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Input a | Input b | Input c | Input d | Output e | Output f | Output 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 |

왼쪽 위에서부터 차례대로 testbench file, design source file, simulation 그리고 진리표이다. Source file을 보면 input a와 b의 XOR 값을 e에, e와 c의 XOR값을 f에, 그리고 f와 d의 XOR값을 g에 할당했고, g가 최종 출력물이 된다. XOR gate의 경우 2개의 입력 값이 00이나 11처럼 서로 같으면 출력 값이 0이되고, 01이나 10처럼 서로 다르면 출력 값으로 1이 반환된다. 따라서 이번에 구현한 회로는 입력 값들 중에서 1의 개수가 짝수이면 출력 값으로 0이 나오고, 홀수이면 출력 값으로 1이 나온다. 이는 simulation의 결과로도 확인할 수 있다. 코드로 구현할 때에는 design source file 사진을 보면 알 수 있듯이 입력 값 2개씩 묶어서 XOR연산을 진행했다. 이 과정을 3번 진행하여 4input 3output XOR연산자를 구현하였다.

**5.**

**텍스트, 스크린샷, 폰트, 영수증이(가) 표시된 사진

자동 생성된 설명텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명**

**스크린샷, 도표, 텍스트, 사각형이(가) 표시된 사진

자동 생성된 설명**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Input a | Input b | Input c | Input d | Output e | Output f | Output 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 |

왼쪽 위에서부터 차례대로 testbench file, design source file, simulation 그리고 진리표이다. AOI는 AND와 OR의 줄인 말이고, Source file을 보면 input a와 b의 AND 값을 e에, e와 c의 AND값을 f에, 그리고 f와 d의 NOR값을 g에 할당했고, g가 최종 출력물이 된다. 코드로 구현할 때에는 design source file 사진을 보면 알 수 있듯이 2개의 input로 AND 연산을 2번 진행하고 마지막에 OR연산을 한 뒤 NOT을 붙였다.

**6.**

(1) NAND gate

도표, 라인이(가) 표시된 사진

자동 생성된 설명

도표, 라인, 기술 도면, 평면도이(가) 표시된 사진

자동 생성된 설명

위의 그림은 Vivado로 그려낸 schematic그림이고, 밑의 그림은 강의자료에서 가져온 schematic그림이다. 강의자료에서는 NAND gate를 AND gate 옆에 작은 동그라미 기호를 붙여서 표현했지만, Vivado에서 만들어진 schematic그림을 보면 AND gate와 NOT gate가 연결 되어있는 것을 확인할 수 있다.

(2) NOR gate

도표이(가) 표시된 사진

자동 생성된 설명

도표, 라인, 기술 도면, 스케치이(가) 표시된 사진

자동 생성된 설명

위의 그림은 Vivado로 그려낸 schematic그림이고, 밑의 그림은 강의자료에서 가져온 schematic그림이다. 강의자료에서는 NOR gate를 OR gate 옆에 작은 동그라미 기호를 붙여서 표현했지만, Vivado에서 만들어진 schematic그림을 보면 OR gate와 NOT gate가 연결 되어있는 것을 확인할 수 있다.

(3) XOR gate

도표, 라인, 그래프, 평면도이(가) 표시된 사진

자동 생성된 설명

도표, 라인, 스케치, 기술 도면이(가) 표시된 사진

자동 생성된 설명

위의 그림은 Vivado로 그려낸 schematic그림이고, 밑의 그림은 강의자료에서 가져온 schematic그림이다. Vivado에서 생성된 schematic그림이 강의자료의 것보다 비교적 복잡해 보이지만 전체적으로 비슷한 양상을 보이고있다.

XOR의 실제 논리식은 이기 때문에 코드를 작성할 때 X^Y말고 (~X&&Y) || (X&&~Y)을 사용해도 되지만 이를 사용하여 코드를 작성하면 schematic 그림은 ^을 사용할 때보다 더 복잡하다.

(4) AOI gate

도표, 라인, 평면도이(가) 표시된 사진

자동 생성된 설명

도표, 스케치, 라인, 기술 도면이(가) 표시된 사진

자동 생성된 설명

위의 그림은 Vivado로 그려낸 schematic그림이고, 밑의 그림은 강의자료에서 가져온 schematic그림이다. Vivado에서 생성된 schematic이 강의자료의 것보다 비교적 복잡해 보이지만 전체적인 양상은 같은 것을 확인할 수 있고, NOR gate를 보면 강의자료에는 OR gate 뒤에 NOT을 의미하는 작은 동그라미 기호로 표현했지만, Vivado의 schematic 그림에서는 OR연산자와 NOT연산자를 연결시켜서 나타낸 것을 볼 수 있다.