4주차 결과보고서

전공: 컴퓨터공학과 학년: 2학년 학번: 20211522 이름: 김정환

**1.**

.................

NAND/NOR/XOR Gate를 Verilog를 통해 다중입력으로 구현하고, Simulation을 통해서 각 Gate의 동작을 확인한다. 그 후 FPGA에 bit stream을 통해 실제 동작을 확인한다.

................

**2.**

.......................

4-input NAND Gate를 구현한 코드는 아래와 같고 NAND의 경우 두 입력 a, b가 있다고 하면, ~(a&b)와 같은 방식으로 Verilog에서 구현하였다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

이 코드를 통해 simulation을 돌린 결과는 아래와 같다.

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

자동 생성된 설명

NAND Gate는 입력이 두 개 모두 1인 경우를 제외하고 1을 출력한다. 이 점과 위의 simulation을 이용해 진리표를 작성하면 아래와 같다.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| In A | In B | In C | In D | Out E | Out F | Out 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 |

이 경우에는 입력이 모두 1인 경우를 제외하고 1을 출력하는 결과를 코드와 같이 중첩하여 계산한 결과를 나타내고 이를 FPGA에서도 똑같이 작동하는 점을 확인할 수 있었다.

.........................

**3.**

.......................

4-input NOR Gate는 아래와 같은 코드로 구현했고, NOR의 경우 두 입력 a, b가 있다고 하면 ~(a|b)와 같은 방식으로 Verilog에서 구현했다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

Simulation의 결과는 아래와 같다.

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

자동 생성된 설명

NOR Gate는 입력 두 가지 모두 0인 경우에만 1을 출력한다. 이외의 경우에는 0을 출력한다. 이 점과 simulation 결과를 통해 진리표를 작성하면 아래와 같다.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| In A | In B | In C | In D | Out E | Out F | Out 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 |

이 경우에는 코드에서 나타냈듯이 두 입력 모두 0인 경우에만 1을 출력하는 결과를 중첩하여 계산한 결과를 나타내고, FPGA에서도 같은 결과를 보임을 확인할 수 있었다.

.........................

**4.**

.......................

4-input XOR Gate의 코드는 아래와 같고 XOR은 두 입력 a, b에 대해서 a^b로 Verilog에서 구현했다.

텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

이를 이용한 simulation의 결과는 아래와 같다.

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

자동 생성된 설명

XOR Gate는 두 가지 입력이 서로 다를 때 1을 출력하고 같을 때 0을 출력한다. 이와 simulation의 결과를 이용하여 진리표를 작성하면 아래와 같다.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| In A | In B | In C | In D | Out E | Out F | Out 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 |

이 경우에는 서로 값이 다른 경우에 1을 출력하는 Gate를 중첩하여 계산한 결과를 나타내며, FPGA에서도 똑같이 작동하는 것을 확인할 수 있었다.

.........................

**5.**

.......................

4-input AOI Gate의 코드는 아래와 같고, AOI Gate는 총 4개의 입력 a, b, c, d를 두 개씩 각각 a&b, c&d로 AND 연산을 진행한 후, 두 출력을 NOR, 즉 ~(e|f)로 구현하였다.

텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

이를 이용한 simulation의 결과는 아래와 같다.

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

자동 생성된 설명

AOI Gate는 각각 두 입력에 대해서 AND 연산 후에 NOR 연산을 진행하는 Gate이다. 이 점과 Simulation의 결과를 이용해 진리표를 작성하면 아래와 같다.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| In A | In B | In C | In D | Out E | Out F | Out 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 |

이 경우에는 두 입력씩 AND 연산을 진행한 후 두 출력에 대해 NOR 연산을 진행한 결과를 보여주며 FPGA에서도 똑 같은 결과를 보여주는 것을 확인할 수 있었다.

.........................

**6.**

.......................

NAND, NOR, XOR, AOI Gate를 4개의 input을 받아서 결과를 출력하는 실험이었다. NAND의 경우 두 입력 a, b가 있다고 하면, ~(a&b)와 같은 방식으로 Verilog에서 구현하였고, 이는 a, b 모두 1인 경우를 제외하고 1을 출력하는 것을 FPGA 보드에서 확인했다. NOR의 경우 두 입력 a, b가 있다고 하면 ~(a|b)와 같은 방식으로 Verilog에서 구현했고, 이는 a, b 모두 0인 경우에만 1을 출력하는 것을 FPGA 보드에서 확인했다. XOR은 두 입력 a, b에 대해서 a^b로 Verilog에서 구현했고, 이는 a, b가 서로 다른 경우 1을 출력하는 점을 FPGA 보드에서 확인했다. AOI Gate는 총 4개의 입력 a, b, c, d를 두 개씩 각각 a&b, c&d로 AND 연산을 진행한 후, 두 출력을 NOR, 즉 ~(e|f)로 구현하였다. 이 경우는 각각 AND 연산이 정상적으로 수행되는 모습, 그리고 이 후, NOR가 잘 수행되는 모습을 FPGA를 이용하여 확인하였다.

.........................

**7.**

.......................

Verilog에서 XOR을 나타낼 때 ^을 이용하여 한 번에 나타낼 수도 있지만 이전에 조사한 것처럼 (~A)B + A(~B)의 식을 이용하여 나타낼 수도 있다. 이는 ((~a)&b)|(a&(~b))와 같이 다른 방식으로도 같은 결과를 나타낼 수 있다.

다음으로는 드모르간의 법칙에 관한 부분이다. 이는 집합의 연산에서도 사용하는 것과 마찬가지로 AND나 OR의 연산의 전체에 NOT을 통해 값을 바꿔준 것은 각각의 입력에 대해 NOT을 한 결과들에 OR과 AND로 바뀌는 법칙에 대한 부분이다. 이를 Verilog에서 사용하는 연산자들로 나타내보면, ~(a&b)는 (~a)|(~b)와 같은 결과를, ~(a|b)는 (~a)&(~b)와 같은 결과를 가진다는 것을 나타낸다.

.........................