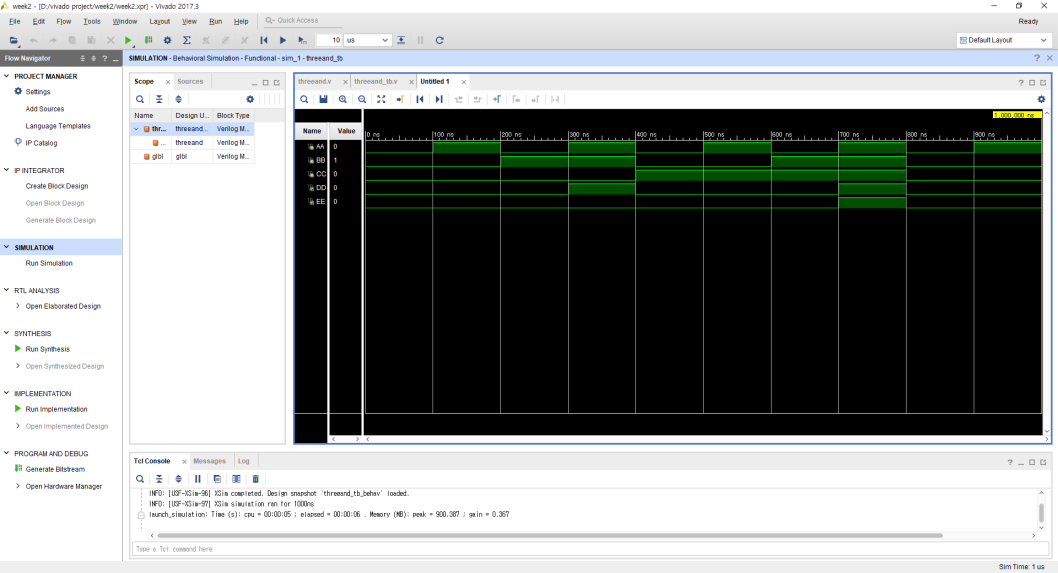
3주차 결과보고서

전공: 수학과 학년: 4학년 학번: 20171273 이름: 심현우

1. **FPGA 동작법을 설명하시오.**

FPGA는 input에 대해 output이 존재하는 논리회로로 input이 논리게이트를 지나면서 결과값인 output을 도출한다. FPGA의 구조는 1주차에 조사하였고 이번 주차에서는 동작법에 대해서 알아본다. FPGA는 HDL을 사용하여 프로그래밍할 수 있는데 우리는 Verilog를 사용하므로 Verilog 기준으로 동작법을 보자. 우선 2, 3주차에 했던 실습처럼 Verilog를 사용하여 프로그래밍을 통해 논리게이트를 작성하고 모듈을 설정한다. 또한 작성한 모듈에 대해 시뮬레이션을 돌릴 수 있다. 하지만 이는 모듈의 동작만 검증할 뿐 실제 하드웨어에서 잘 작동하는지 알 수 없다. 따라서 FPGA를 컴퓨터와 연결한 후 하드웨어에서의 동작을 확인해줘야 한다. 동작 확인을 위한 Verilog에서의 과정은 다음과 같다. 우선 device설정이 잘 되어있는지부터 확인한다. 검증할 하드웨어와 assignment를 한 후, 논리게이트의 input과 output으로 연결할 pin을 assignment한다. 이때, input은 프로그래밍했던 Verilog의 모듈과, 즉 소프트웨어와 연결되어 있고, output은 하드웨어에서 눈으로 확인할 수 있는 LED등과 연결되어 있다. Input pin과 output pin을 verilog의 코드 중 각각의 port와 set\_property를 통해 연결하고 I/O port에서 잘 확인이 된다면 pin assignment는 완료한 것이다. Verilog 모듈 자체에서 synthesis와 implement를 확인했던 것처럼 확인하고 연결된 device의 동작을 확인하면 된다.

1. **3-input AND gate의 simulation 결과 및 과정에 대해서 설명하시오.**

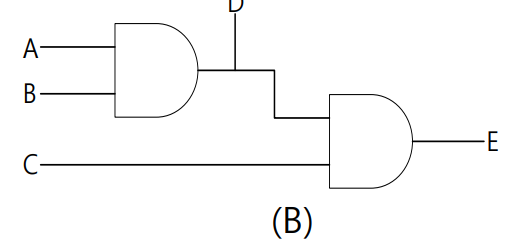


테이블이(가) 표시된 사진

자동 생성된 설명텍스트, 영수증, 스크린샷이(가) 표시된 사진

자동 생성된 설명

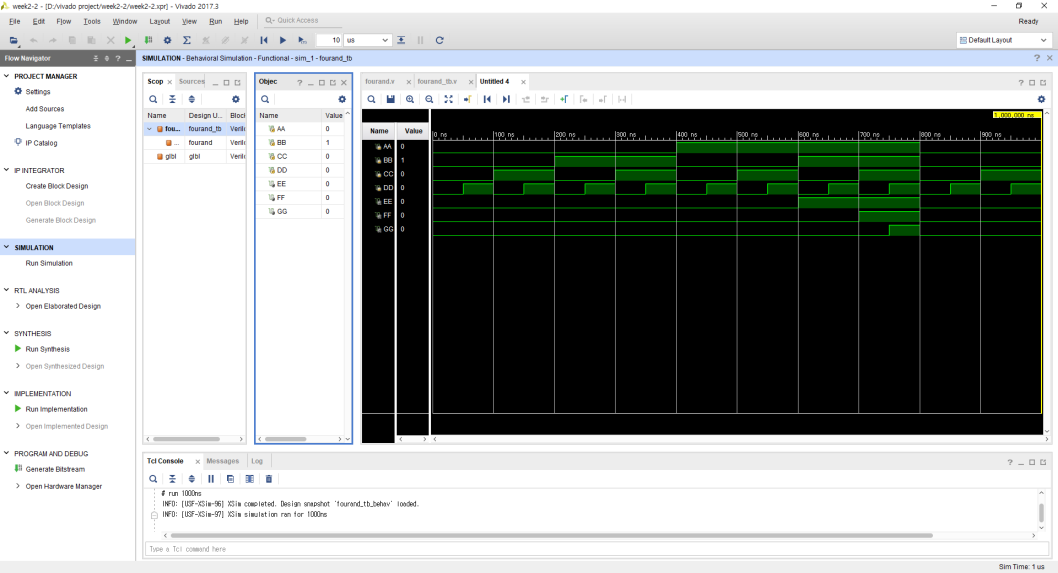
3-input AND 게이트의 simulation 결과는 위와 같다. 밑에 코드를 보게 되면 input으로 A,B,C 세가지가 있고, output으로 E하나가 있다. 이때, D는 inout으로 input과 output으로 동시에 쓰일 수 있다. 밑에 assign을 보게 되면 D는 A와 B가 AND 게이트를 지나 결과로 나온 output이고 이것을 다시 input으로 하여 D와 C가 AND 게이트를 지나 결과로 나온 ouput이 E가 된다. 따라서 이는 강의자료에서 (B)에 해당하는 AND 게이트를 나타낸다. AND 게이트는 input 모두 1의 값을 가질 때 output이 1로 나온다.



옆의 simulation을 위한 시간값을 보면, A는 100ns, B는 200ns, C는 400ns마다 신호를 주고 있는 것을 확인할 수 있다. 이렇게 설정한 이유는 총 1000ns에서 100,200,400으로 설정할 시 input의 모든 경우의 수 8가지를 확인할 수 있기 때문이다. 이는 위의 simulation 사진을 보면 확인할 수 있다. 처음 0ns에서는 A,B,C에서 모두 신호가 없어 0의 값을 갖는다. 이때 논리게이트에 의하면 D는 A&B이므로 역시 0이 되고 E는 D&C이므로 역시 0이 된다. 다음 100ns를 보면 A가 1로 신호가 바뀐 것을 알 수 있다. 따라서 이를 0ns부터 진리표로 작성하게 되면 다음과 같다.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| In A | In B | In C | Out D | Out E |
| 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 |

1. **4-input AND gate의 simulation 결과 및 과정에 대해서 설명하시오.**

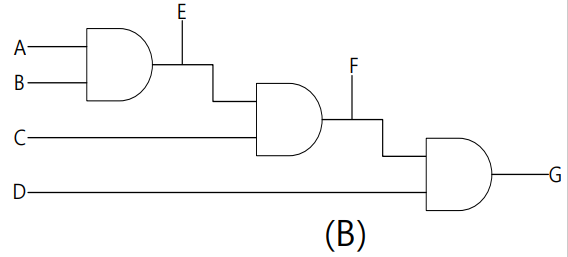


테이블이(가) 표시된 사진

자동 생성된 설명테이블이(가) 표시된 사진

자동 생성된 설명

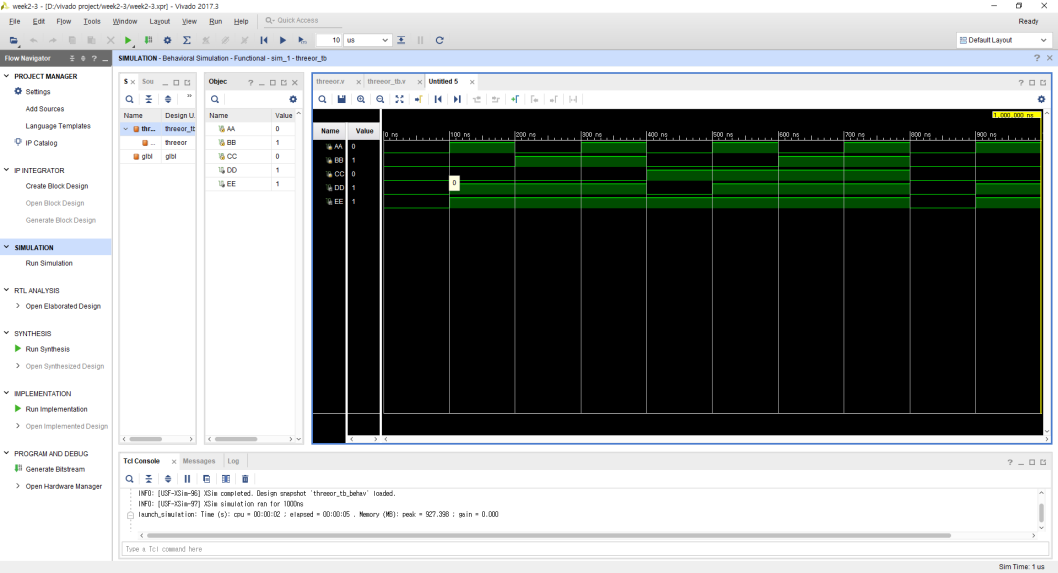
4-input AND 게이트의 simulation 결과는 위와 같다. 밑에 코드를 보게 되면 input으로 A,B,C,D 네가지가 있고, output으로 G 하나가 있다. 이때, E,F는 inout으로 input과 output으로 동시에 쓰일 수 있다. 밑에 assign을 보게 되면 E는 A와 B가 AND 게이트를 지나 결과로 나온 output이고 이것을 다시 input으로 하여 E와 C가 AND 게이트를 지나 결과로 나온 ouput이 F가 된다. 또 다시 F를 input으로 F&D를 해서 나온 결과 output이 G가 된다. 따라서 이는 강의자료에서 (B)에 해당하는 AND 게이트를 나타낸다. AND 게이트는 input 모두 1의 값을 가질 때 output이 1로 나온다.

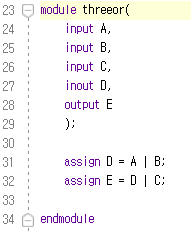


옆의 simulation을 위한 시간값을 보면, A는 400ns, B는 200ns, C는 100ns, D는 50ns마다 신호를 주고 있는 것을 확인할 수 있다. 이렇게 설정한 이유는 총 1000ns에서 50,100,200,400으로 설정할 시 input의 모든 경우의 수 16가지를 확인할 수 있기 때문이다. 이는 위의 simulation 사진을 보면 확인할 수 있다. 처음 0ns에서는 A,B,C,D에서 모두 신호가 없어 0의 값을 갖는다. 이때 논리게이트에 의하면 E는 A&B이므로 역시 0이 되고 F는 E&C이므로 역시 0이 된다. 마지막 결과값 역시 G=F&D이므로 G도 0이다. 다음 50ns를 보면 D가 1로 신호가 바뀐 것을 알 수 있다. 마지막 케이스 750ns를 보면 A,B,C,D가 모두 1인 것을 확인할 수 있다. 이따 AND게이트에 따라 E=A&B에서 E는 1이 되고 F=E&C에서 F도 1 이 된다. 마지막으로 G=F&D로 G도 1이 된다. 따라서 이를 0ns부터 진리표로 작성하게 되면 다음과 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 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 |

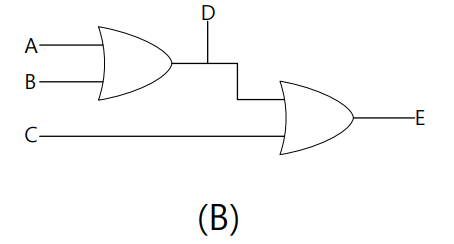
1. **3-input OR gate의 simulation 결과 및 과정에 대해서 설명하시오.**



 텍스트, 영수증, 스크린샷이(가) 표시된 사진

자동 생성된 설명

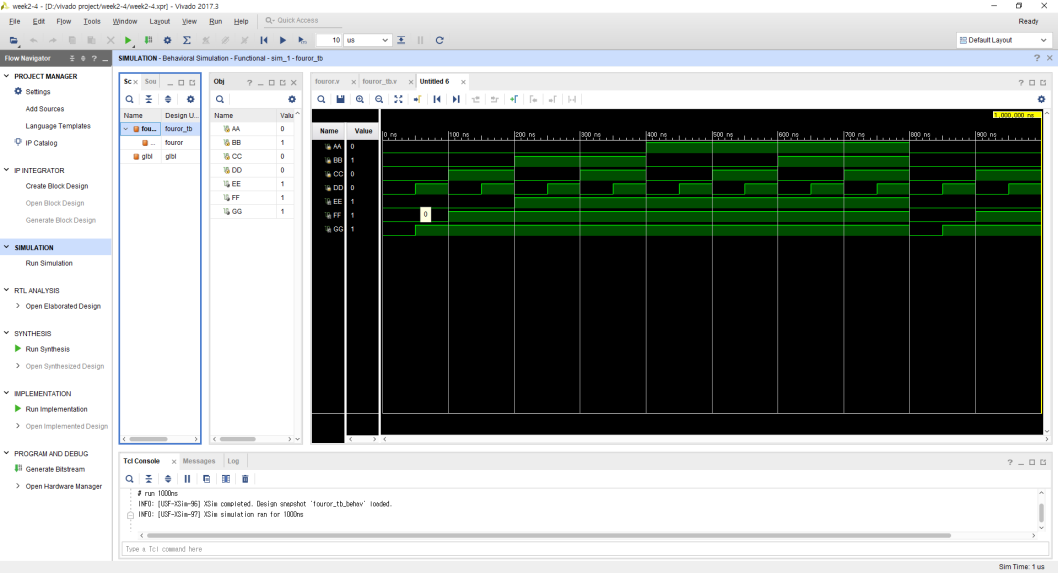
3-input OR 게이트의 simulation 결과는 위와 같다. 밑에 코드를 보게 되면 input으로 A,B,C 세가지가 있고, output으로 E하나가 있다. 이때, D는 inout으로 input과 output으로 동시에 쓰일 수 있다. 밑에 assign을 보게 되면 D는 A와 B가 OR 게이트를 지나 결과로 나온 output이고 이것을 다시 input으로 하여 D와 C가 OR 게이트를 지나 결과로 나온 ouput이 E가 된다. 따라서 이는 강의자료에서 (B)에 해당하는 OR 게이트를 나타낸다. OR 게이트는 input값 중 적어도 하나가 1의 값을 가질 때 output이 1로 나온다.



옆의 simulation을 위한 시간값을 보면, A는 100ns, B는 200ns, C는 400ns마다 신호를 주고 있는 것을 확인할 수 있다. 이렇게 설정한 이유는 총 1000ns에서 100,200,400으로 설정할 시 input의 모든 경우의 수 8가지를 확인할 수 있기 때문이다. 이는 위의 simulation 사진을 보면 확인할 수 있다. 처음 0ns에서는 A,B,C에서 모두 신호가 없어 0의 값을 갖는다. 이때 논리게이트에 의하면 D는 A|B이므로 역시 0이 되고 E는 D|C이므로 역시 0이 된다. 다음 100ns를 보면 A가 1로 신호가 바뀐 것을 알 수 있다. 이 때, 바뀐 신호 A에 따라 D의 값이 1로 변한 것을 알 수 있다. 이는 A와 B가 OR게이트를 지나기 때문이다. 따라서 이를 0ns부터 진리표로 작성하게 되면 다음과 같다.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| In A | In B | In C | Out D | Out E |
| 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 |

1. **4-input OR gate의 simulation 결과 및 과정에 대해서 설명하시오.**

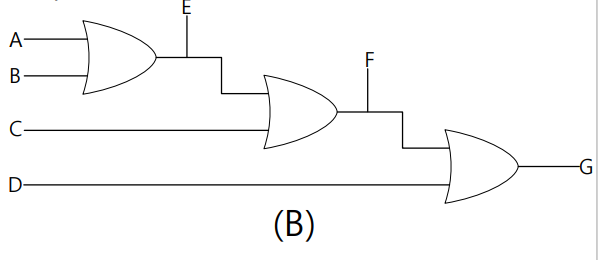


텍스트, 영수증, 스크린샷이(가) 표시된 사진

자동 생성된 설명 테이블이(가) 표시된 사진

자동 생성된 설명

4-input OR 게이트의 simulation 결과는 위와 같다. 밑에 코드를 보게 되면 input으로 A,B,C,D 네가지가 있고, output으로 G 하나가 있다. 이때, E,F는 inout으로 input과 output으로 동시에 쓰일 수 있다. 밑에 assign을 보게 되면 E는 A와 B가 OR 게이트를 지나 결과로 나온 output이고 이것을 다시 input으로 하여 E와 C가 OR 게이트를 지나 결과로 나온 ouput이 F가 된다. 또 다시 F를 input으로 F|D를 해서 나온 결과 output이 G가 된다. 따라서 이는 강의자료에서 (B)에 해당하는 OR 게이트를 나타낸다. OR 게이트는 input값 중 적어도 하나가 1의 값을 가질 때 output이 1로 나온다.

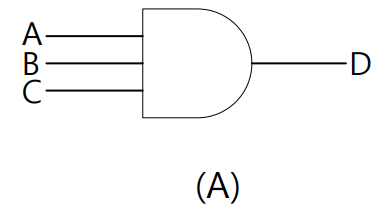


옆의 simulation을 위한 시간값을 보면, A는 400ns, B는 200ns, C는 100ns, D는 50ns마다 신호를 주고 있는 것을 확인할 수 있다. 이렇게 설정한 이유는 총 1000ns에서 50,100,200,400으로 설정할 시 input의 모든 경우의 수 16가지를 확인할 수 있기 때문이다. 이는 위의 simulation 사진을 보면 확인할 수 있다. 처음 0ns에서는 A,B,C,D에서 모두 신호가 없어 0의 값을 갖는다. 이때 논리게이트에 의하면 E는 A|B이므로 역시 0이 되고 F는 E|C이므로 역시 0이 된다. 마지막 결과값 역시 G=F|D이므로 G도 0이다. 다음 50ns를 보면 D가 1로 신호가 바뀐 것을 알 수 있다. 이 때, D의 변화에 따라 G의 값도 1이 된다. 이는 OR게이트의 동작 방법 때문이다. 따라서 이를 0ns부터 진리표로 작성하게 되면 다음과 같다.

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

1. **결과 검토 및 논의사항.**

위의 나온 결과들을 Boolean 식을 통해 직접 계산해 봤을 때 오류 없이 잘 나온 것을 확인할 수 있다. 논의사항으로 강의자료에서 게이트가 (A), (B)로 두개가 있는데 이를 비교해보자. 위의 게이트들에서 inout 없이 모든 input을 하나의 논리게이트로 결과를 도출하는 것이 (A)이다. 예를 들어 3-input AND게이트는 다음과 같이 생겼다.



이 게이트에 대한 진리표를 작성해 보면 다음과 같다.

|  |  |  |  |
| --- | --- | --- | --- |
| In A | In B | In C | In D |
| 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 1 | 1 | 1 |

이는 위의 2번에서와 결과값이 같다. AND게이트의 경우 모든 input이 1이 되어야 output이 1이 되기 때문이다.

OR의 경우도 보자. 4-input OR 게이트의 경우 4가지의 input이 하나의 OR게이트를 통과하는 것을 생각해보자. 이를 진리표로 나타내면 다음과 같다.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| In A | In B | In C | In D | In E |
| 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 0 | 1 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 |

이 또한 위의 5번의 4-input OR게이트와 진리표가 같다. 이는 OR게이트의 특성 Input중 적어도 하나가 1이면 output이 1이기 때문이다.

1. **추가 이론 조사 및 작성.**

Verilog 코드 작성 시 timescale 1ns/1ps를 맨 위에서 볼 수 있다. 이는 프로그래밍에서 단위 시간과 precision을 나타낸다. 먼저 1ns는 단위로 신호가 유지되는 시간의 단위를 나타낸다. 밑에서 #100으로 코드를 작성한다면 이는 100ns동안 신호가 유지되는 것을 의미한다. 뒤의 1ps는 precision으로 신호를 줄 때 딜레이, 즉 어느정도 정확하게 신호값을 주느냐를 나타낸다.