4주차 결과보고서

전공: 아트앤테크놀로지 학년: 3학년 학번: 20191098 이름: 백승주

**1.**

4 input 3 output NAND, NOR, XOR, AOI 게이트를 소스코드로 구현하고 testbench 소스로 시뮬레이션 결과를 확인하며 진리표를 작성한다.

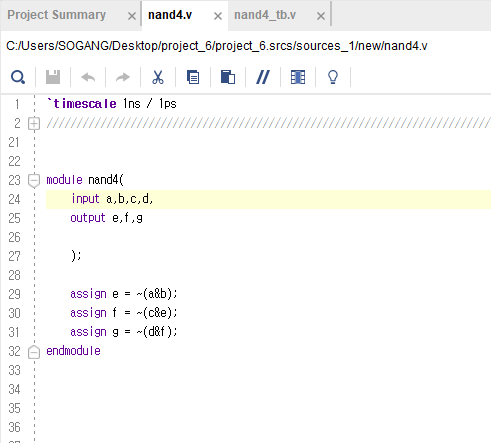
-FPGA를 통해 구현한 코드의 동작을 확인한다.

-결과를 보며 4 input 1 output NAND, NOR XOR 게이트와의 차이점을 조사한다.

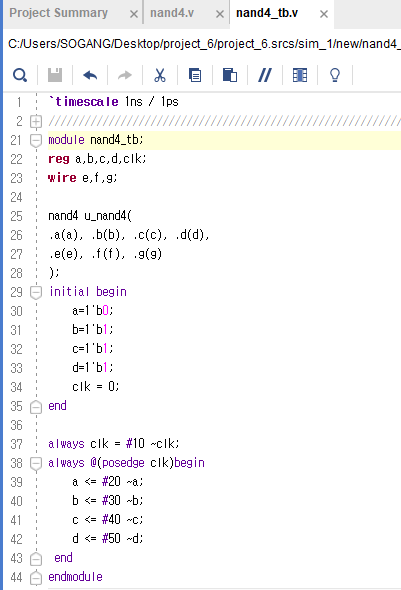
**2. 4-input NAND gate**



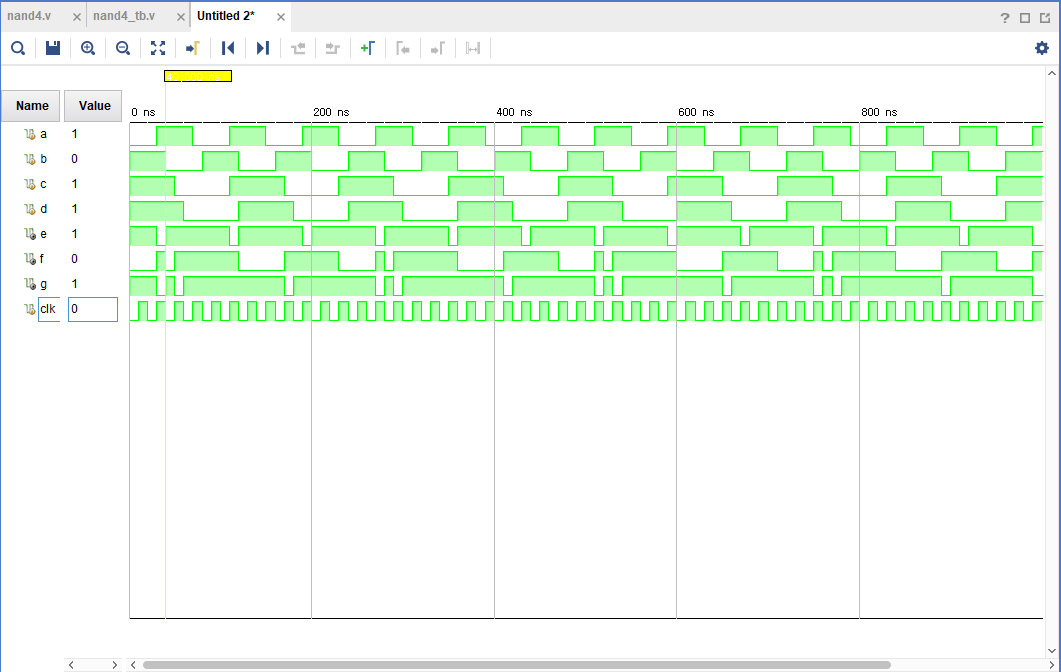
먼저 2개의 입력을 받는 NAND gate는 AND 게이트에 not을 취한 형태이다. 위의 4-input NAND gate는 A, B, C, D 라는 4개의 input을 가지고 3개의 output을 가지는 형태의 게이트다. 이 게이트는 먼저 A, B의 입력 값을 NAND 게이트로 받아 결과값인 E를 출력하고 이 output E를 다시 C와 함께 input으로 받아 결과값인 output F를 출력한다. 그리고 마지막으로 이 F와 D를 입력을 받아 최종 결과값인 G를 출력한다. 이에 대한 Verilog코드는 다음과 같다.



assign문을 활용해 output e, f ,g에 NAND 게이트의 결과값을 할당했다. 또한 코드의 결과를 simulation하기 위한 testbench코드는 다음과 같다.



이 코드에 의한 시뮬레이션 결과는 clk 값이 positive edge일 때마다 input값들이 각자 다른 시간 간격을 두고 바뀌며 4 input Nand gate의 모든 input a, b, c, d 의 모든 경우의 수와 이에 따른 output들의 다음과 같은 시뮬레이션 결과값을 얻을 수 있을 것이다.



이에 따른 진리표는 다음과 같다.

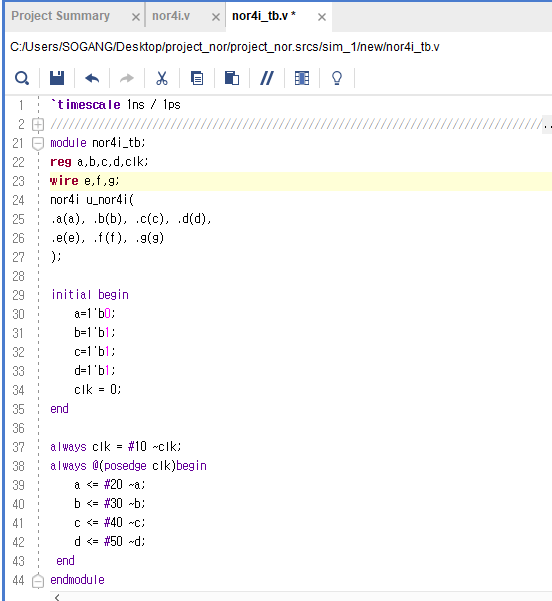
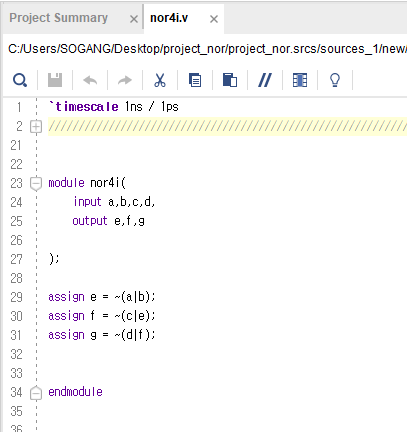
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Input A** | **Input B** | **Input C** | **Input D** | **Output E** | **Output F** | **Output G** |
| 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| 0 | 0 | 0 | 1 | 1 | 1 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 1 | 0 | 1 |
| 0 | 1 | 0 | 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 |

진리표의 결과를 보면 입력 값들이 모두 1일 때 뿐만 아니라 다른 때에도 최종 결과값인 G가 0을 출력하는 것을 알 수 있다. 즉, 4 input NAND 게이트는 AND게이트를 합쳐 만들어 4개가 모두 1일 때만 1을 출력하는 4 input AND 게이트에 단순히 NOT을 취한 결과와는 다른 것을 알 수 있다.

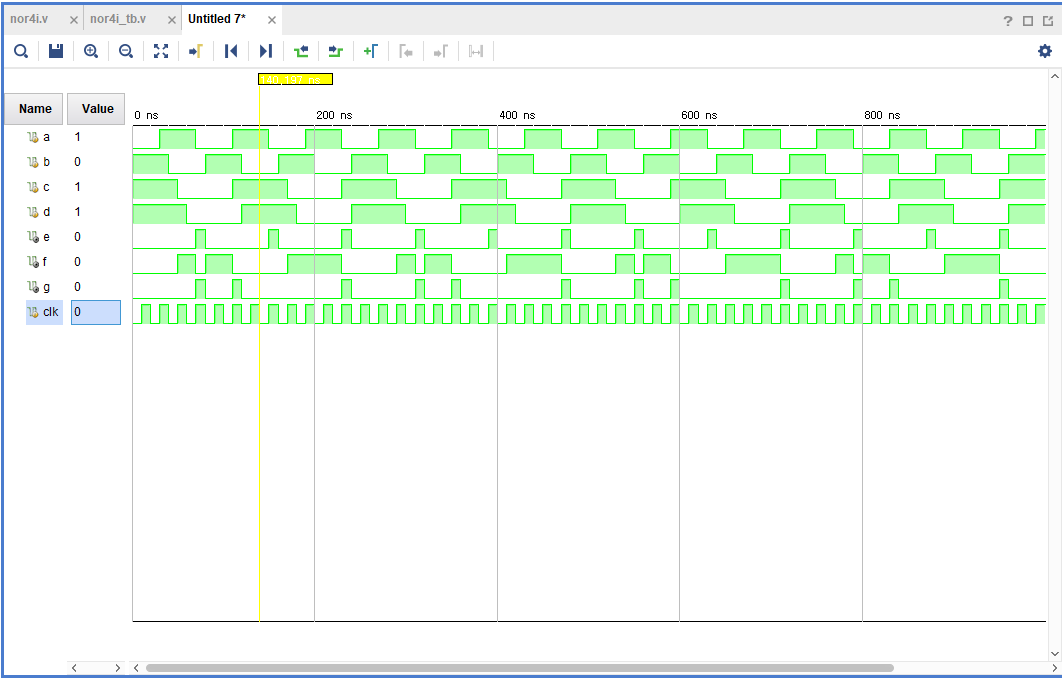
**3. 4- input Nor gate**



먼저 Nor gate는 or gate에 not을 취한 형태이다. 4-input Nor gate는 A, B, C, D 라는 4개의 input을 가지고 3개의 output을 가지는 게이트로 이 게이트는 먼저 A, B 의 input 값을 NOR 게이트로 받고 output인 E를 출력한다. output E를 다시 C와 함께 input으로 받아 output F를 출력하고 이 output F,D를 받아 최종 결과값인 G를 출력한다.



Nor gate의 source코드와 testbench 코드는 위와 같다.



그 결과 위와 같은 시뮬레이션 결과가 나오고 이를 토대로 진리표를 만들면 다음과 같다.

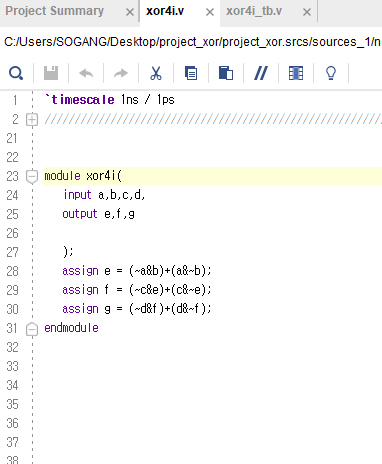
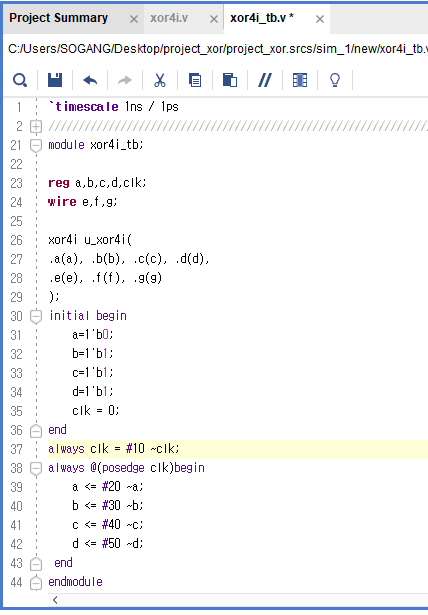
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Input A** | **Input B** | **Input C** | **Input D** | **Output E** | **Output F** | **Output G** |
| 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 0 | 1 | 0 | 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 |

진리표의 결과를 보면 입력 값들이 모두 0일 때 뿐만 아니라 다른 때에도 최종 결과값인 G가 1을 출력하는 것을 알 수 있다. 즉, 4 input NOR 게이트는 OR게이트를 합쳐 만들어 4개가 모두 1일 때만 1을 출력하는 4 input AND 게이트에 단순히 NOT을 취한 결과와는 다른 것을 알 수 있다.

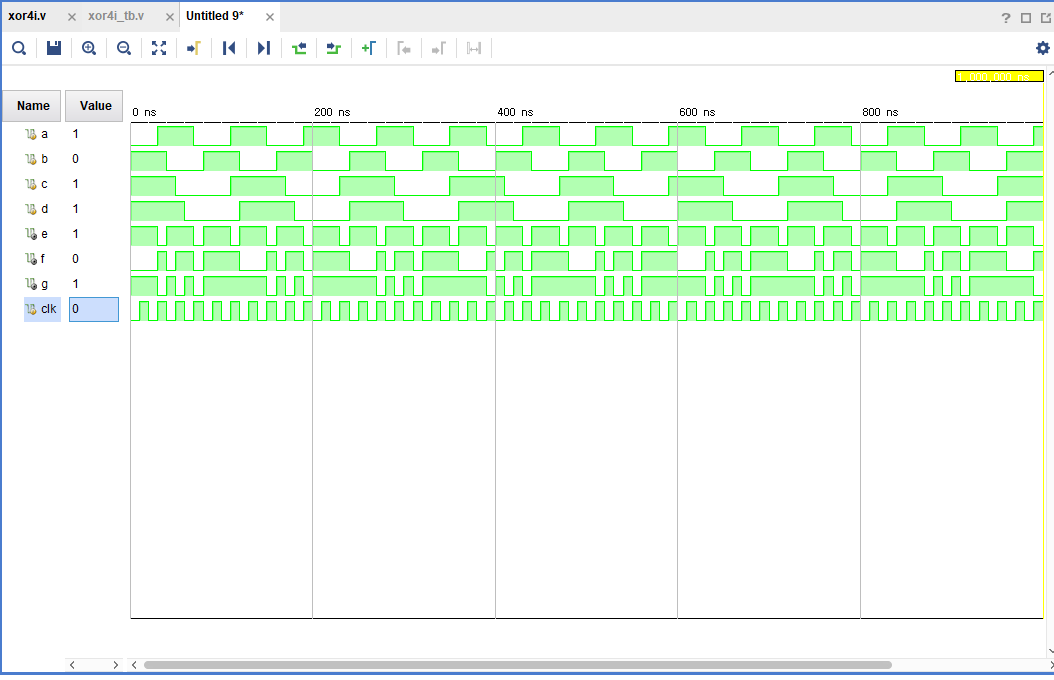
**4.**  **4- input Xor gate**



Xor은 exclusive or 을 의미하며 4이 -input Xor gate는 A, B, C, D 라는 4개의 input 값들을 XOR 게이트로 순차적으로 받고 그 결과값들도 입력값으로 받아 최종 결과값으로 G를 출력한다.



위 코드들의 실행 결과 다음과 같은 시뮬레이션이 출력된다.



이를 토대로 진리표를 작성하면 다음과 같다.

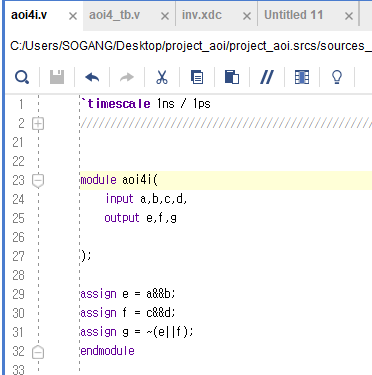
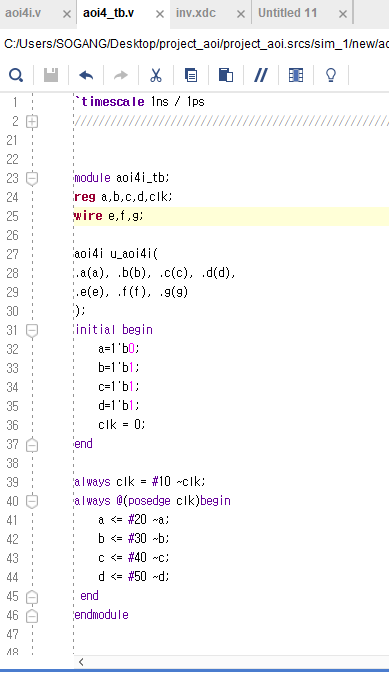
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Input A** | **Input B** | **Input C** | **Input D** | **Output E** | **Output F** | **Output G** |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 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 |

진리표의 결과를 보면 4 input Xor 게이트는 입력 값의 수가 짝수면 결과값으로 0을 출력하고 입력 값 개수가 홀수면 1을 출력하는 것을 알 수 있다.

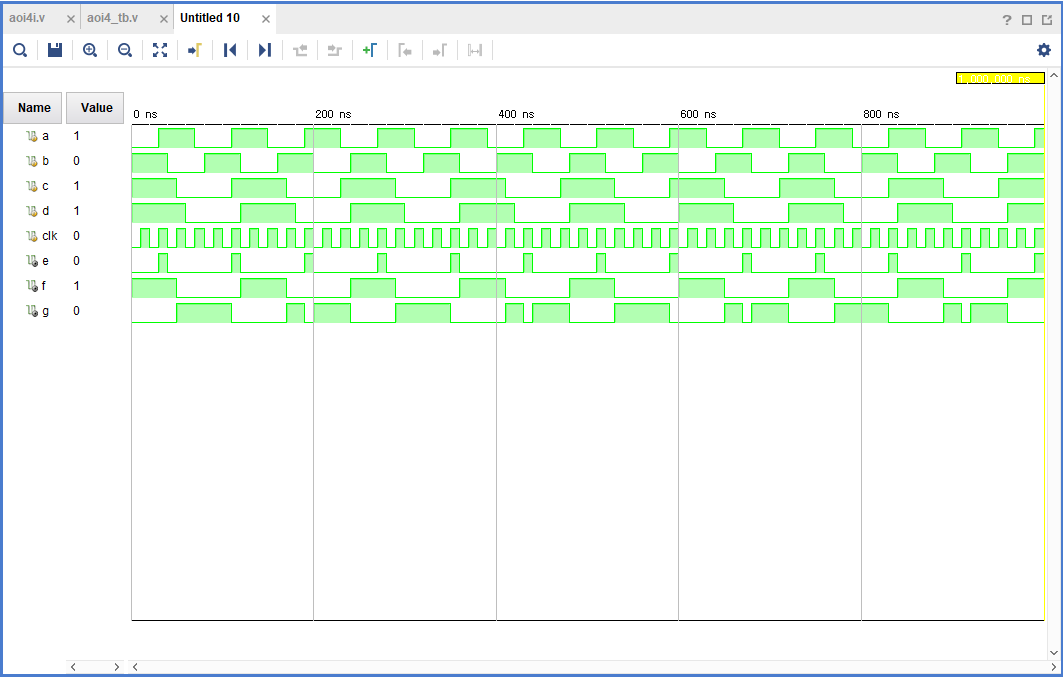
**5. 4- input AOI gate**



위의 4-input OR gate는 4-input AOI 게이트는 B, C, D 라는 4개의 input을 가지고 3개의 output을 가지는 게이트다.. 먼저 A, B의 입력 값을 AND게이트로 받고 결과값인 E를 출력한다. 또한 마찬가지로 C,D를 AND 게이트의 입력 값으로 받고 그 결과 값으로 F를 출력한다. 그리고 이러한 E, F 를 OR 게이트의 입력 값으로 받아 최종 결과값인 G를 출력한다.

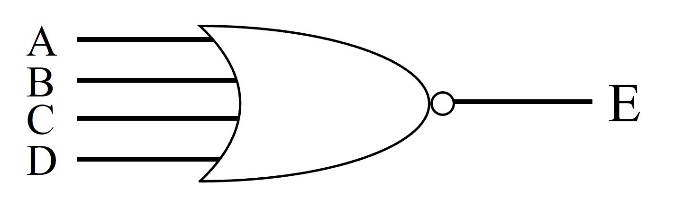
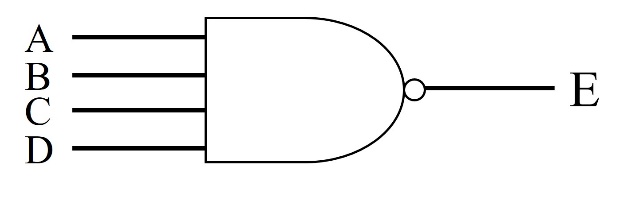
코드들에 따른 시뮬레이션은 다음과 같다.



시뮬레이션 결과를 토대로 진리표를 만들면 다음과 같다.

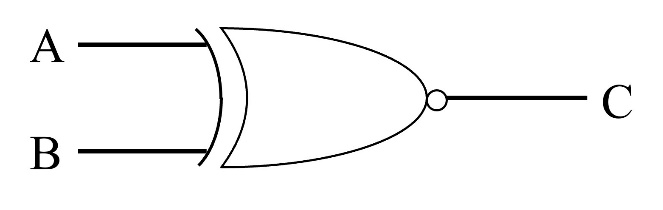
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Input A** | **Input B** | **Input C** | **Input D** | **Output E** | **Output F** | **Output G** |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 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 |

**6.**



위 그림들은 4주차 강의자료 (A) 에 해당하는 4 input 1 output NAND 게이트와 4 input 1 outpu t Nor 게이트를 나타낸 것이다. 위 회로들의 입력 값에 따른 결과값은 각각 4 input AND 게이트와 4 input Or 게이트에 not을 취한 것과 동일하다. 즉, 4 input 3 output과 4 input 1 output 의 결과가 동일했던 3주차의 AND 게이트와 OR 게이트와 달리 이번 NAND 게이트와 NOR 게이트는 4 input 3 output 일 때와 4 input 3 output 일 때의 결과가 다른 것을 알 수 있다. 반면에 Xor 게이트는 3 input과 1 input의 차이가 발생하지 않는다. 따라서 실제 회로를 설계할 때 이 점을 주의하며 설계해야 할 것이다.

**7.** XNOR 게이트



위 회로는 XNOR 게이트의 그림이다. XNOR 게이트는 exclusive nor을 의미한다. 이 게이트의 결과값은 XOR 게이트에 NOT을 취한 것과 같다. 즉 다시 말해 두 입력 값이 동일하면 1, 서로 다르면 0을 출력한다. 이를 수식으로 나타내면 다음과 같다.

(X⊕Y)’ = XY + X’Y’