주차 보고서

전공: 수학과 학년: 4학년 학번: 20171273 이름: 심현우

1. **실험 목적**

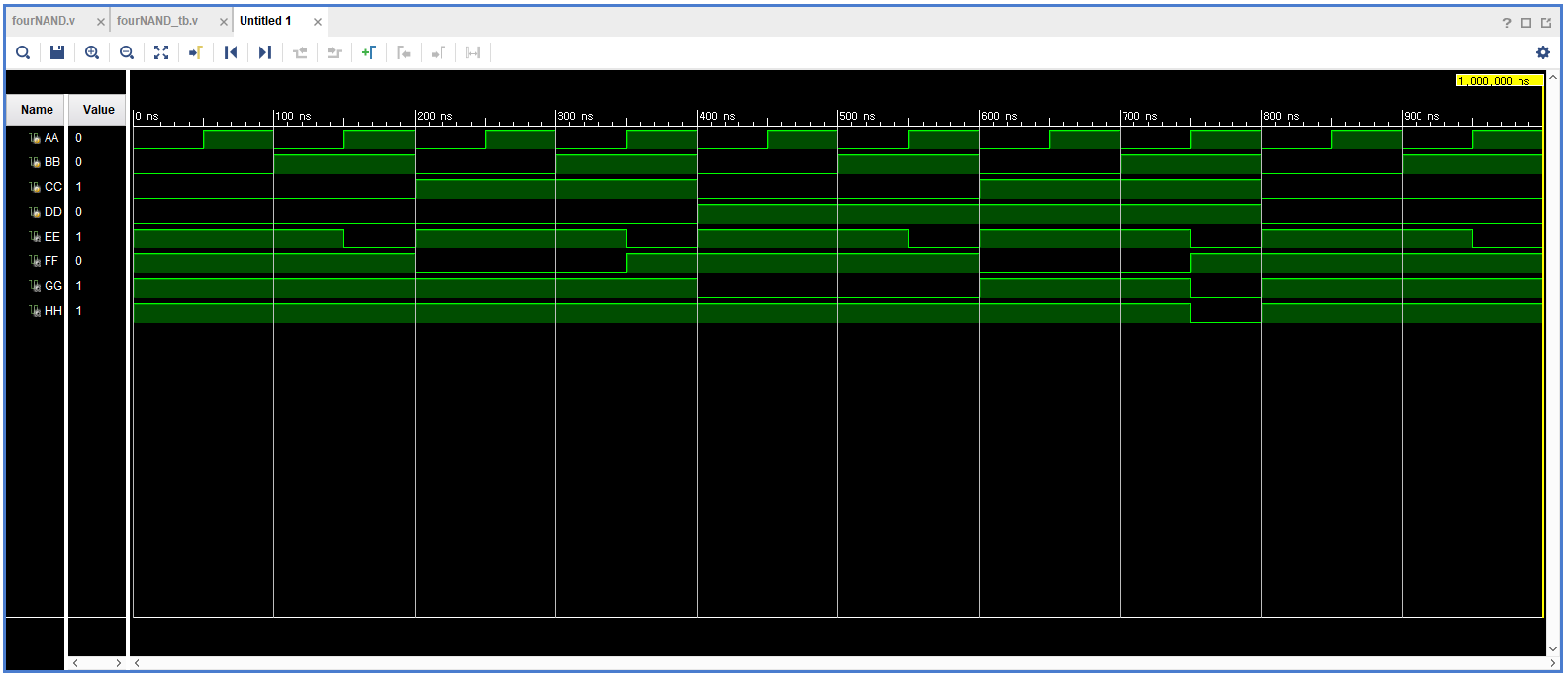
회로에서 자주 쓰이는 NAND, NOR, XOR, AOI게이트에 대해서 구현하고 truth table을 확인하는 실험을 진행했다. 다중 입력, 즉 input값이 많은 상황에 대해서 같은 결과를 출력하기 위해 여러가지 회로가 존재할 수 있는데, 조금 더 간단한 회로를 구성하기 위해 이 NAND, NOR, XOR, AOI가 많이 쓰인다. 기본적인 회로 AND, OR, NOT과 함께 NAND, NOR, XOR, AOI를 조합하여 회로를 간단하게 구성하고 집적도를 높여 전력 감소와 제조 비용에 이점을 가져올 수 있다.

1. **4-input NAND gate의 simulation 결과 및 과정에 대해 설명하시오.**

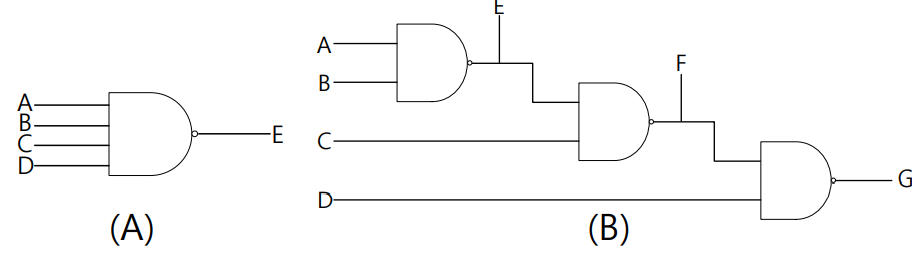
테이블이(가) 표시된 사진

자동 생성된 설명테이블이(가) 표시된 사진

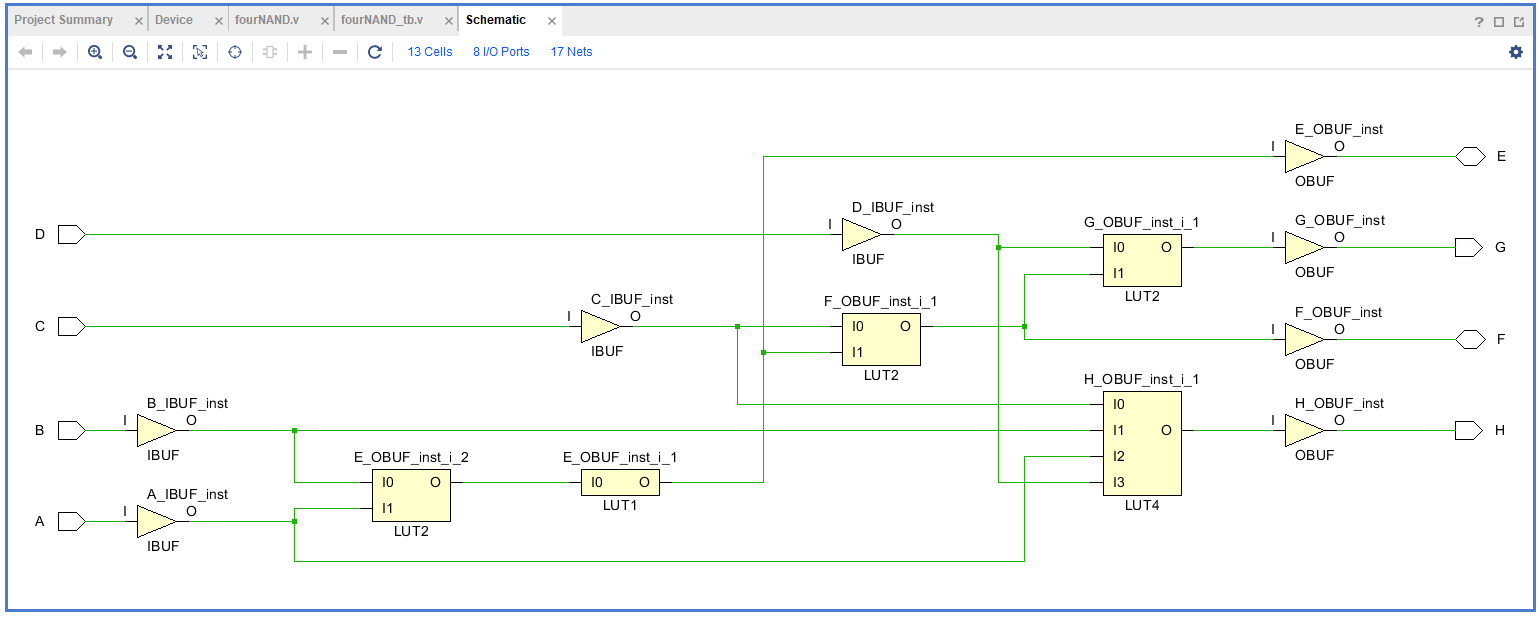
자동 생성된 설명



위의 코드부터 살펴보자. 4-input NAND게이트을 구성했다. 위의 A, B, C, D가 input으로 4개가 들어갔고 H는 (A)의 회로를 구성하기 위한 output이고, E, F, G는 (B)의 회로를 구성하기 위한 inout과 output이다. Assign을 확인해보고 schematic과 함께 확인해보자.



구성하고자 하는 schematic은 위와 같았다. 따라서 이를 구성하기 위해 assign 코드를 구성했다. 우선 H=~(A&B&C&D)로 (A)를 구성하기 위해 4가지 input을 AND로 하고 마지막으로 NOT을 했다. (B)는 inout을 이용하여 E=~(A&B), F=~(E&C), G=~(F&D)로 하여 구성했다. Vivado 프로그램 내에서 확인한 schematic은 아래와 같다.



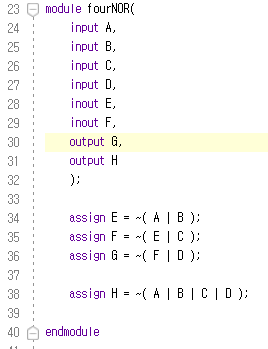
이를 H와 G로 나눠서 보게 되면 H의 input에 A, B, C, D가 있는 것을 확인할 수 있다. 이는 LUT4를 통과하고 BUFF를 통과한다. G를 보게 되면 F와 D가 input으로 들어오는 것을 확인할 수 있고, F는 C와 E를, E는 A와 B를 input으로 하는 것을 확인할 수 있다.

simulation결과로 truth table를 작성해보자.

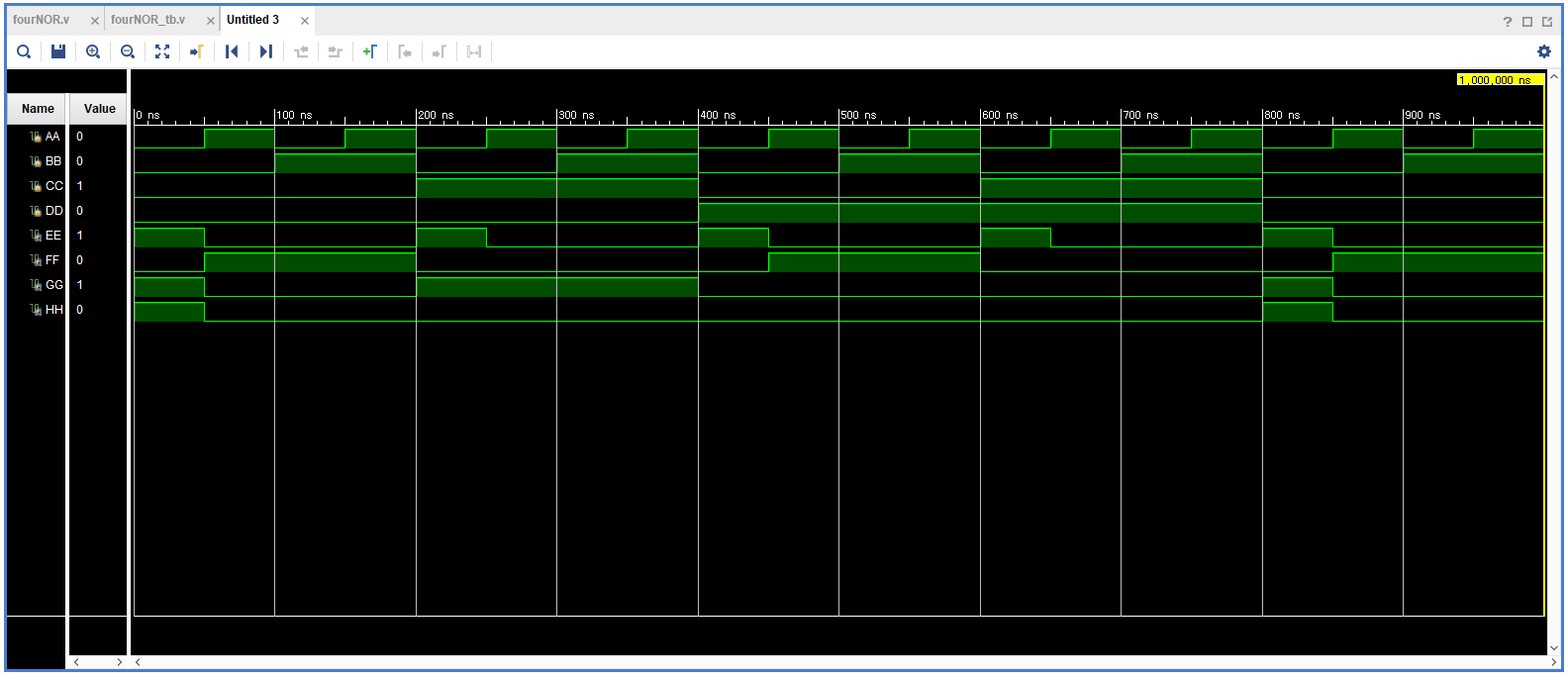
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| In A | In B | In C | In D | Out E | Out F | Out G | Out H |
| 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 |
| 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 |
| 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 |

위에서 확인할 수 있듯이 빨간색 부분은 (A)와 (B)의 결과, G와 H가 다른 것을 확인할 수 있다. 이는 G의 input중 D가 1일 때, 나머지 값들이 0이어도 F가 1이 될 수 있어 결과가 0이 된다. 하지만 H의 경우 모두가 1이어야 결과가 0이 된다.

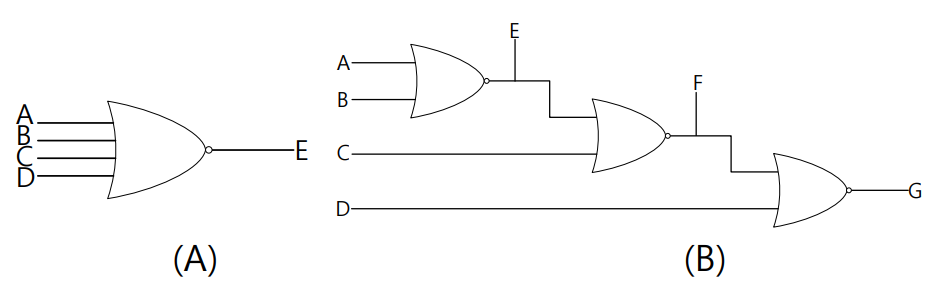
1. **4-input NOR gate의 simulation 결과 및 과정에 대해 설명하시오.**

테이블이(가) 표시된 사진

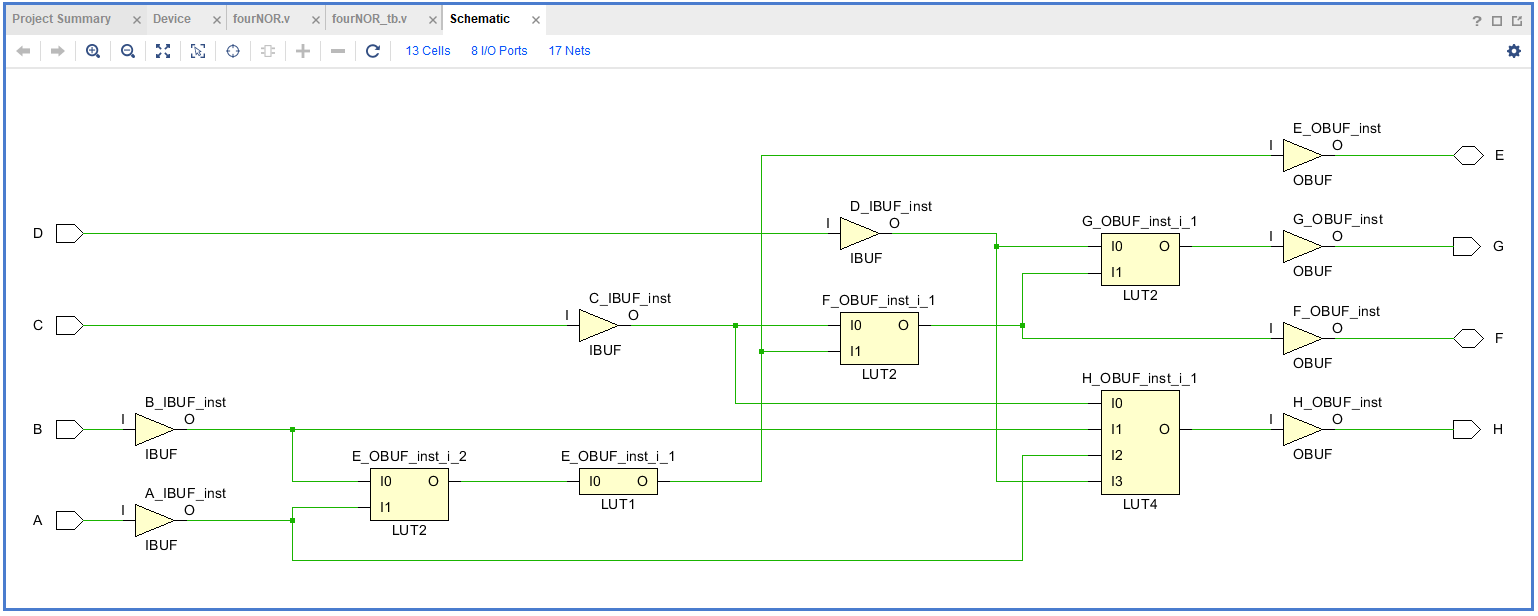
자동 생성된 설명



위의 코드부터 살펴보자. 4-input NOR게이트을 구성했다. 위의 A, B, C, D가 input으로 4개가 들어갔고 H는 (A)의 회로를 구성하기 위한 output이고, E, F, G는 (B)의 회로를 구성하기 위한 inout과 output이다. Assign을 확인해보고 schematic과 함께 확인해보자.



구성하고자 하는 schematic은 위와 같았다. 따라서 이를 구성하기 위해 assign 코드를 구성했다. 우선 H=~(A|B|C|D)로 (A)를 구성하기 위해 4가지 input을 OR로 하고 마지막으로 NOT을 했다. (B)는 inout을 이용하여 E=~(A|B), F=~(E|C), G=~(F|D)로 하여 구성했다. Vivado 프로그램 내에서 확인한 schematic은 아래와 같다.



이를 H와 G로 나눠서 보게 되면 H의 input에 A, B, C, D가 있는 것을 확인할 수 있다. 이는 LUT4를 통과하고 BUFF를 통과한다. G를 보게 되면 F와 D가 input으로 들어오는 것을 확인할 수 있고, F는 C와 E를, E는 A와 B를 input으로 하는 것을 확인할 수 있다.

simulation결과로 truth table를 작성해보자.

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

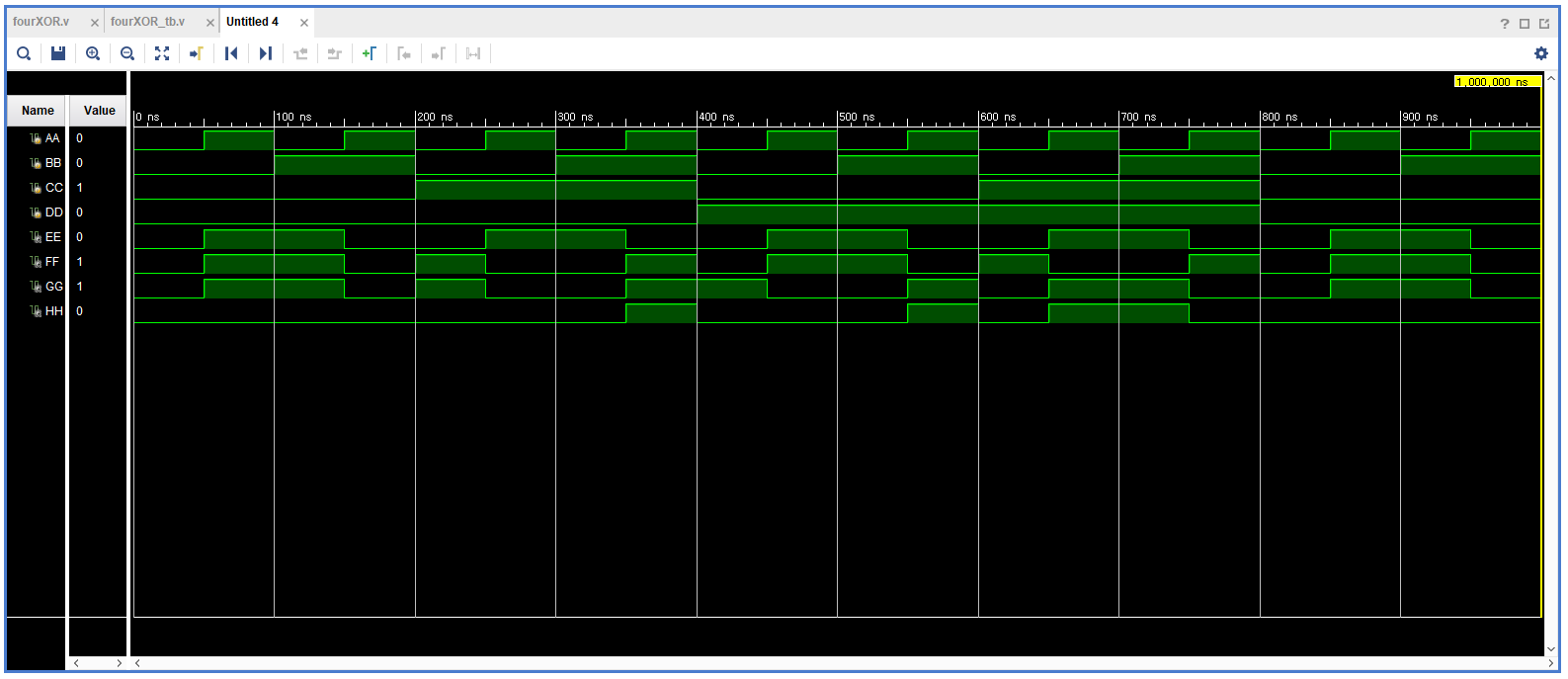
위에서 확인할 수 있듯이 빨간색 부분은 (A)와 (B)의 결과, G와 H가 다른 것을 확인할 수 있다. 이는 G의 input중 D가 0일 때, 나머지 값들이 1이어도 F가 0이 될 수 있어 결과가 1이 된다. 하지만 H의 경우 모두가 0이어야 결과가 1이 된다.

1. **4-input XOR gate의 simulation 결과 및 과정에 대해 설명하시오.**

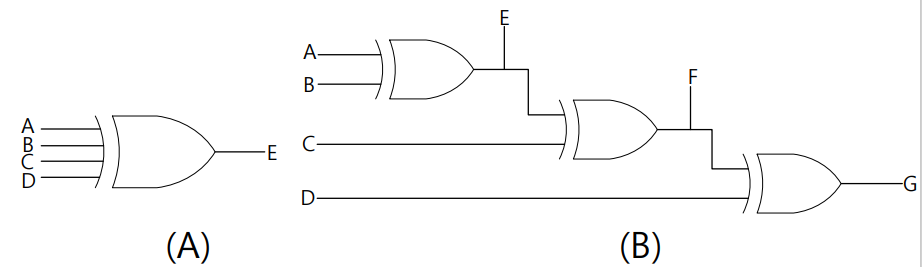
텍스트이(가) 표시된 사진

자동 생성된 설명 테이블이(가) 표시된 사진

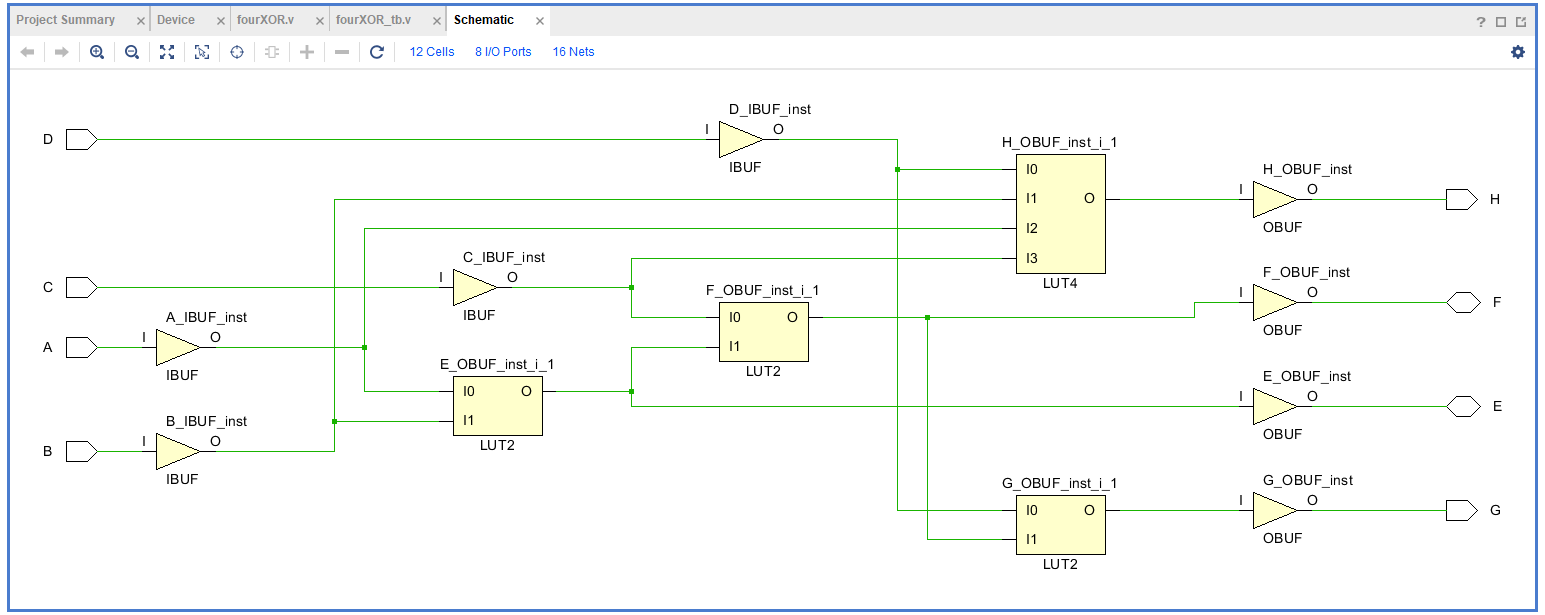
자동 생성된 설명



위의 코드부터 살펴보자. 4-input XOR게이트을 구성했다. 위의 A, B, C, D가 input으로 4개가 들어갔고 H는 (A)의 회로를 구성하기 위한 output이고, E, F, G는 (B)의 회로를 구성하기 위한 inout과 output이다. Assign을 확인해보고 schematic과 함께 확인해보자.



구성하고자 하는 schematic은 위와 같았다. 따라서 이를 구성하기 위해 assign 코드를 구성했다. 우선 H=( ~A & B & C & D ) | ( A & ~B & C & D ) | ( A & B & ~C & D ) | ( A & B & C & ~D )로 (A)를 구성하기 위해 4가지 input을 XOR로 했다. (B)는 inout을 이용하여 E=~(A|B), F=~(E|C), G=~(F|D)로 하여 구성했다. Vivado 프로그램 내에서 확인한 schematic은 아래와 같다.



이를 H와 G로 나눠서 보게 되면 H의 input에 A, B, C, D가 있는 것을 확인할 수 있다. 이는 LUT4를 통과한다. G를 보게 되면 F와 D가 input으로 들어오는 것을 확인할 수 있고, F는 C와 E를, E는 A와 B를 input으로 하는 것을 확인할 수 있다.

simulation결과로 truth table를 작성해보자.

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

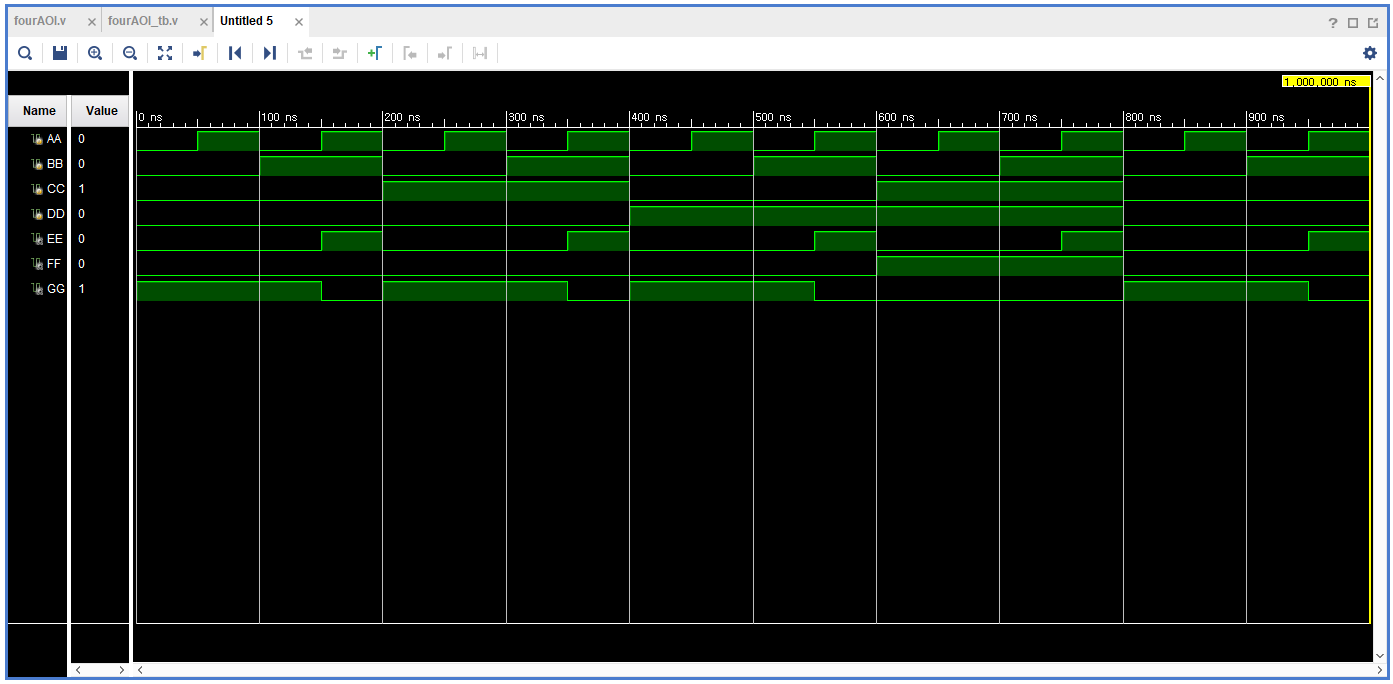
위에서 확인할 수 있듯이 빨간색 부분은 (A)와 (B)의 결과, G와 H가 다른 것을 확인할 수 있다. H의 결과는 A, B, C, D의 input중 1의 개수가 짝수이어야 하지만, G의 경우 D와 F를 비교하기 때문이다.

1. **4-input AOI gate의 simulation 결과 및 과정에 대해 설명하시오.**

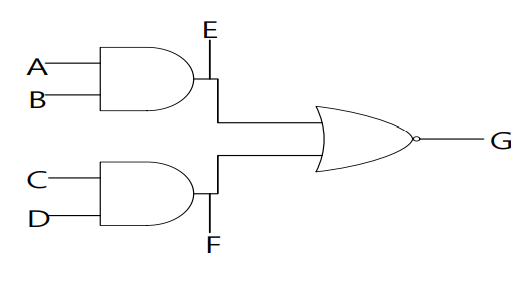
테이블이(가) 표시된 사진

자동 생성된 설명 테이블이(가) 표시된 사진

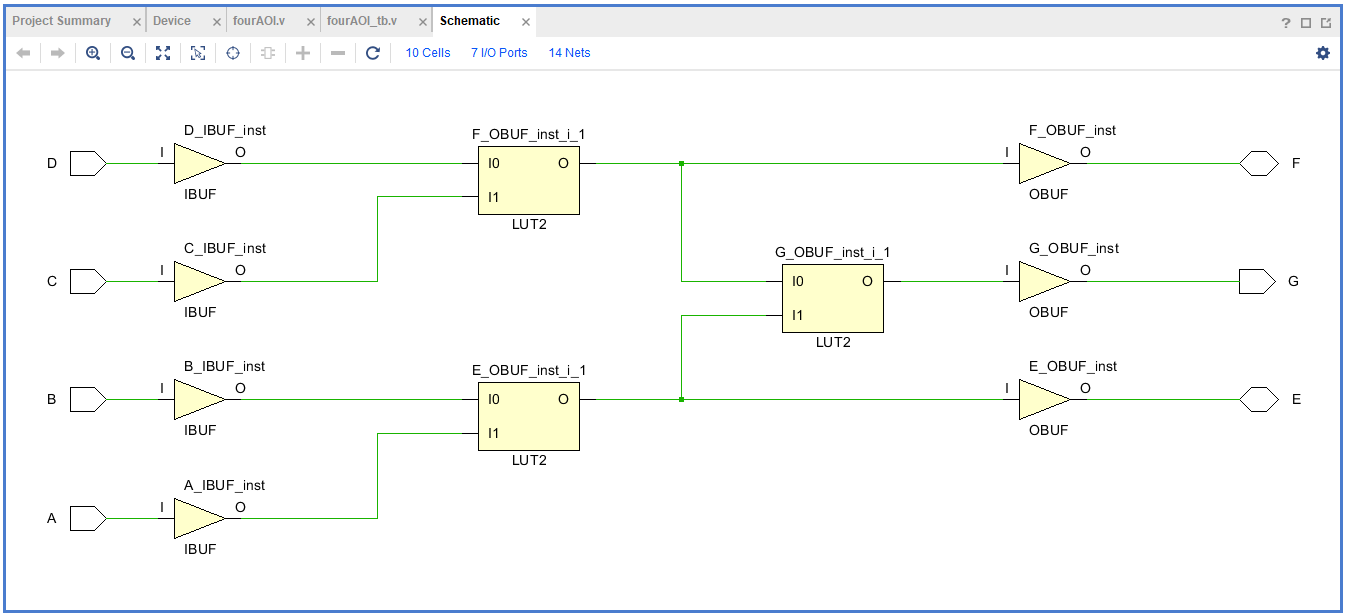
자동 생성된 설명



위의 코드부터 살펴보자. 4-input AOI게이트을 구성했다. 위의 A, B, C, D가 input으로 4개가 들어갔고 E, F는 inout으로 E는 A, B의 ouput역할을, F는 C, D의 output역할을 하며 E,F는 마지막으로 G의 input으로 들어간다. Assign을 확인해보고 schematic과 함께 확인해보자.



구성하고자 하는 schematic은 위와 같았다. 따라서 이를 구성하기 위해 assign 코드를 구성했다. G는 inout E와 F를 이용하여 G=~(E|F)로 하고 E와 F는 E=A&B, F=C&D로 했다. Vivado 프로그램 내에서 확인한 schematic은 아래와 같다.



G를 보게 되면 E와 F가 input으로 들어오는 것을 확인할 수 있고, E는 A와 B를, F는 A와 B를 input으로 하는 것을 확인할 수 있다.

simulation결과로 truth table를 작성해보자.

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

1. **결과검토 및 논의사항**

스스로 계산하여 얻은 결과와 vivado 프로그램으로 얻은 simulation을 통해 확인한 truth table은 다른 것 없이 같았다. 결과가 잘 나온 것을 확인할 수 있었다. Schematic 또한 구현하고자 했던 회로대로 이상 없이 나왔다. 논의사항으로는 NAND만 사용하거나 NOR만 사용하는 즉, 논리게이트의 구성이 하나여도 이를 중복하거나 inout을 사용하는 등 여러가지 방법에 의하여 다양한 결과를 나타나게 할 수 있다는 점이다.

1. **추가이론 조사 및 작성**

NAND, NOR, XOR, AOI 이외 XNOR 회로도 있다. XNOR은 XOR 회로에서 NOT을 추가한 것으로 XOR의 반대이다. XOR은 input이 서로 다를 때, 1의 값을 출력하지만 XNOR은 input이 서로 같을 때, 1의 값을 출력한다. Input이 여러 개인 경우 input의 1의 값이 짝수 개 일 때, 1의 값을 결과로 출력한다.