3주차 결과보고서

|  |
| --- |
| - AND/OR/NOT Gate의 동작 이해 및 확인  - 3 input AND Gate 동작 이해 및 확인  - 4 input AND Gate 동작 이해 및 확인  - 3 input OR Gate 동작 이해 및 확인  - 4 input OR Gate 동작 이해 및 확인  - 결과검토 및 논의사항  - 추가이론 조사 |

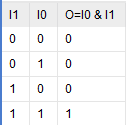
20141196 김성희

1. AND/OR/NOT Gate 동작 이해 및 확인

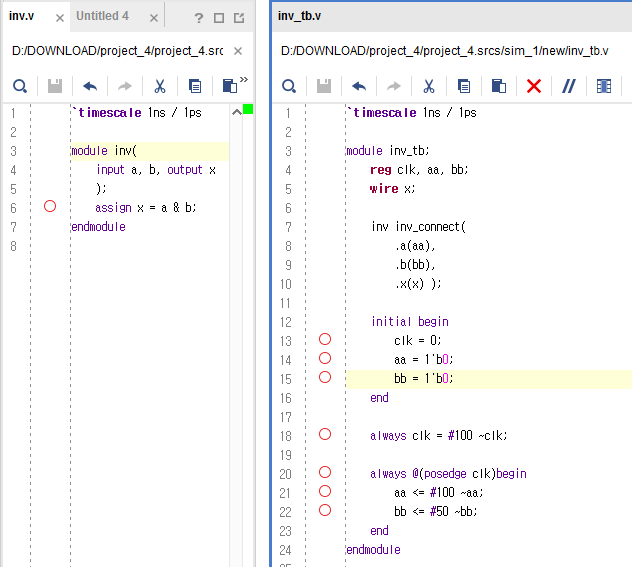
**1-1. AND Gate**

* 2개의 입력 값으로 1개의 출력 값을 도출한다. 코드, schematic, Simulation, 진리표는 아래와 같다. Output = input1 & input2

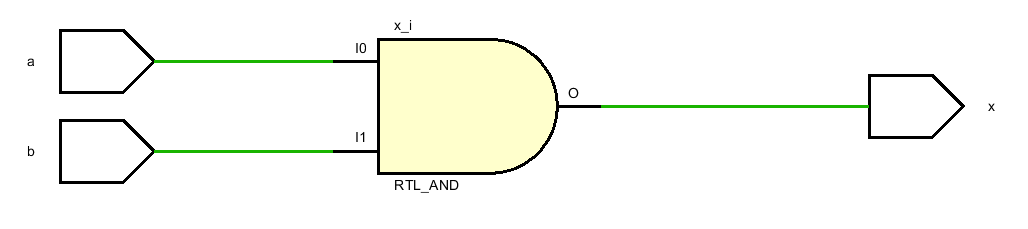
**진리표**

****

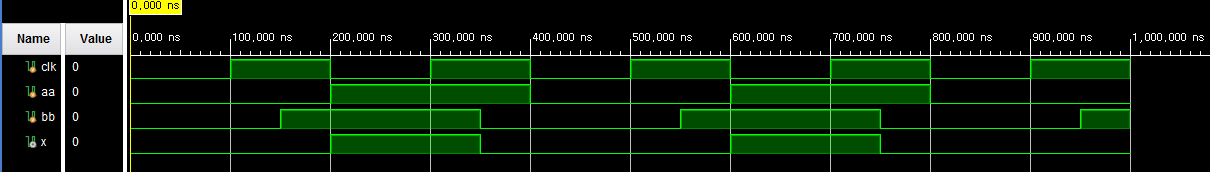
**코드**

****

**Schematic**

****

**Simulation**

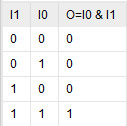
****

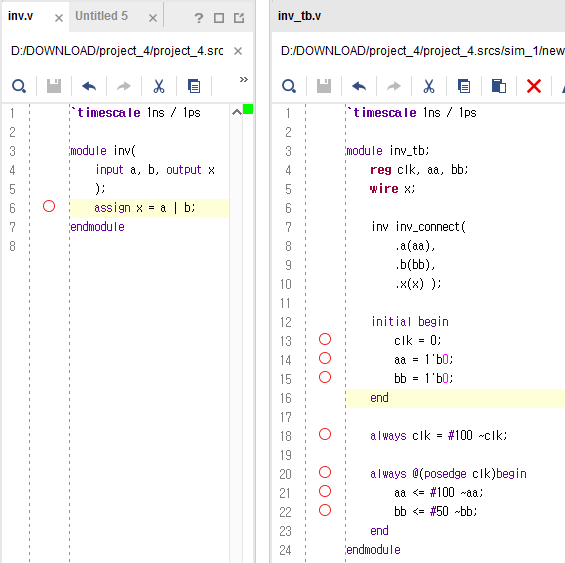
aa와 bb가 input이고 x가 output이다. aa와 bb 모두 1(직사각형)인 경우 x값 또한 1(직사각형)이다. (나머지의 경우 x=0(선)) clk의 riging edge에서부터 100ns 후에 aa값이 반전 되고 50ns 후에 bb값이 반전 된다.

**1-2. OR Gate**

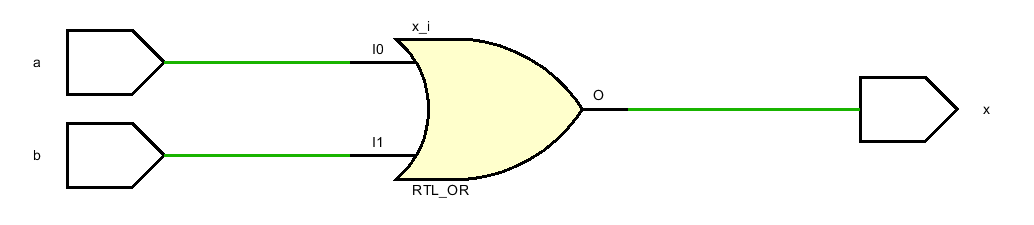
* 2개의 입력 값으로 1개의 출력 값을 도출한다. 코드, schematic, Simulation, 진리표는 아래와 같다. Output = input1 | input2

**진리표**

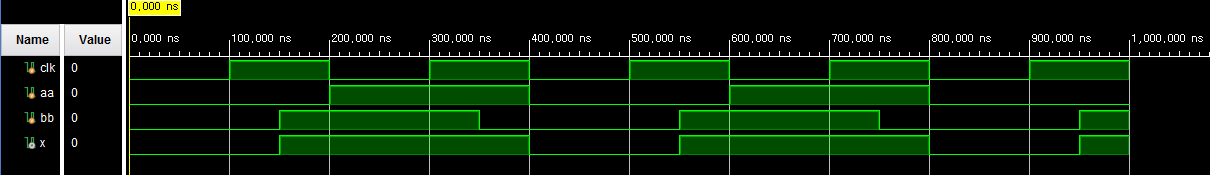
****

**코드**

**Schematic**

****

**Simulation**

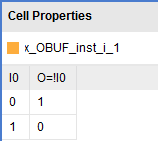
****

aa와 bb가 input이고 x가 output이다. aa와 bb 모두 0(선)인 경우 x값 또한 0(선)이다. (나머지의 경우 x=1(직사각형)) clk의 riging edge에서부터 100ns 후에 aa값이 반전 되고 50ns 후에 bb값이 반전 된다.

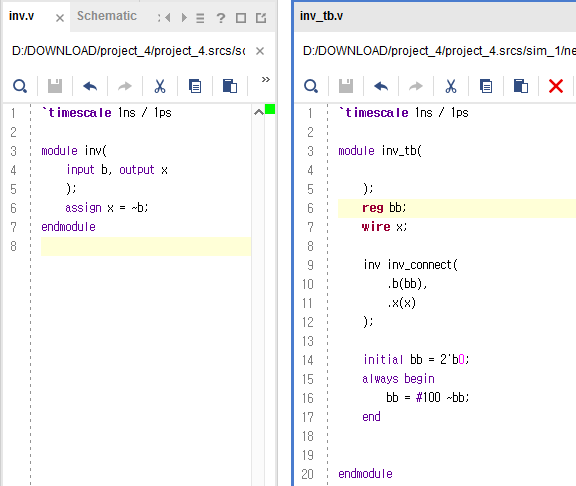
**1-3. NOT Gate**

* 1개의 입력 값으로 1개의 출력 값을 도출한다. 코드, schematic, Simulation, 진리표는 아래와 같다. Output = ~input

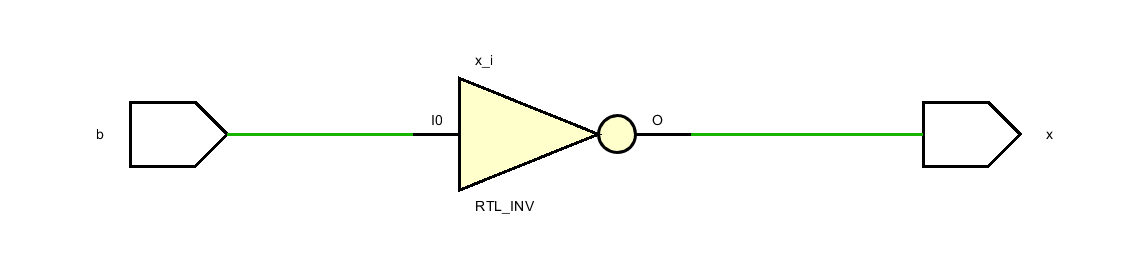
**진리표**

****

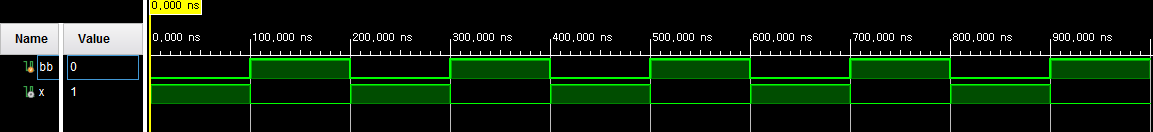
**코드**

****

**Schematic**

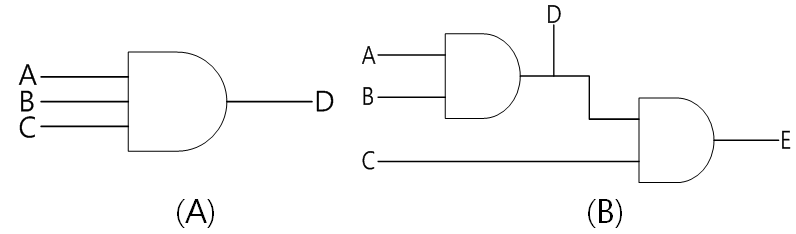
****

**Simulation**

****

2. 3 input AND Gate 동작 이해 및 확인

X

****

Y

X

**- A: 3inputs 1output, B: 3inputs 2outputs**

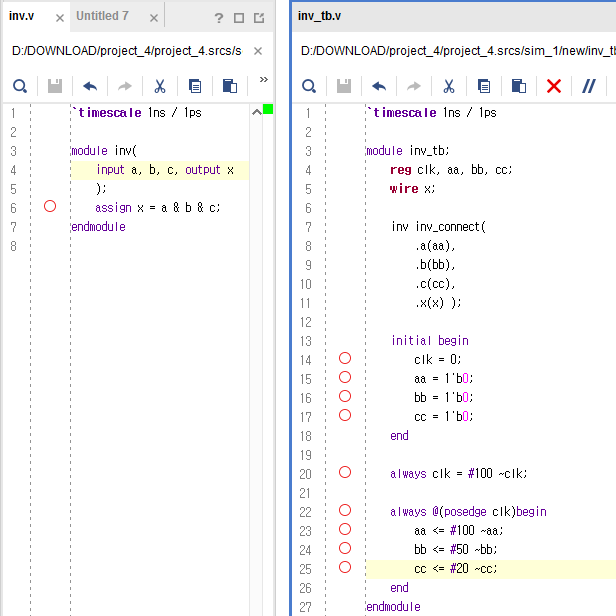
**2-1. Boolean 비교**

**(A) X = A & B & C**

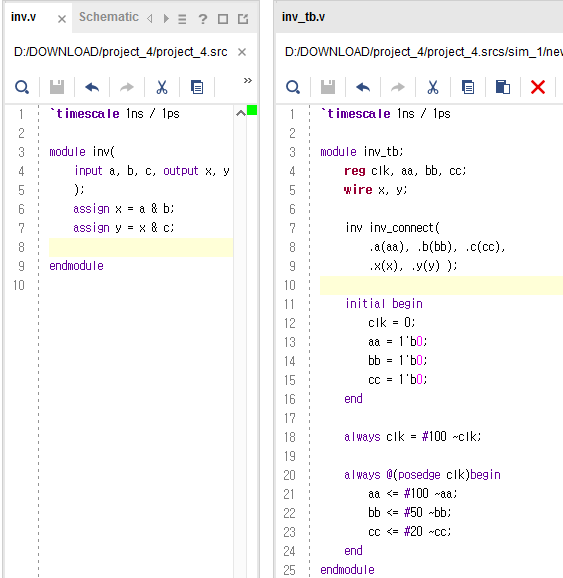
**(B) X = A & B, Y = X & C [즉 Y = (A & B) & C]**

**2-2. 코드**

**(A) 1output**

****

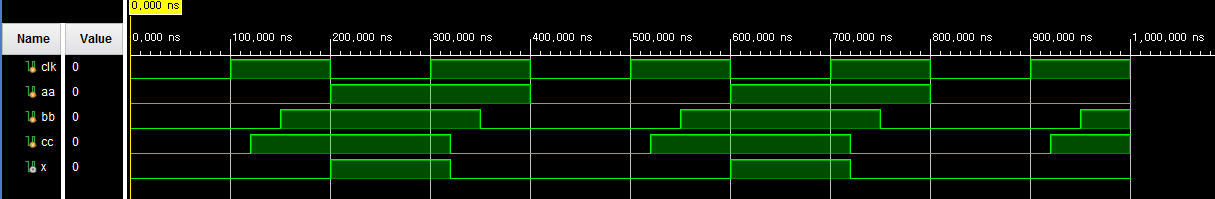
**(B) 2outputs**

****

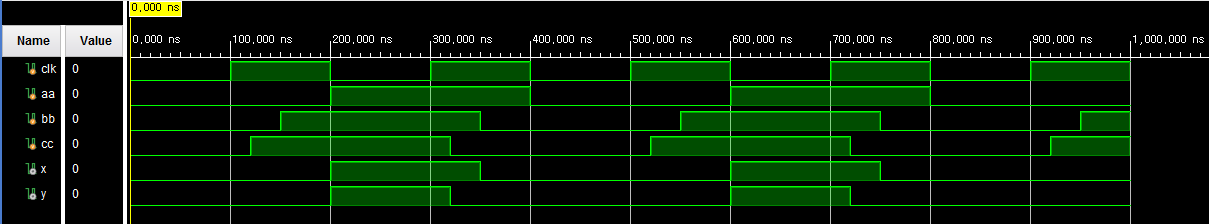
**A와 달리 B는 output이 두 개이며 최종 output을 두 번에 걸쳐서 구한다.**

**2-3. simulation**

**(A) 1output**



**(B) 2outputs**

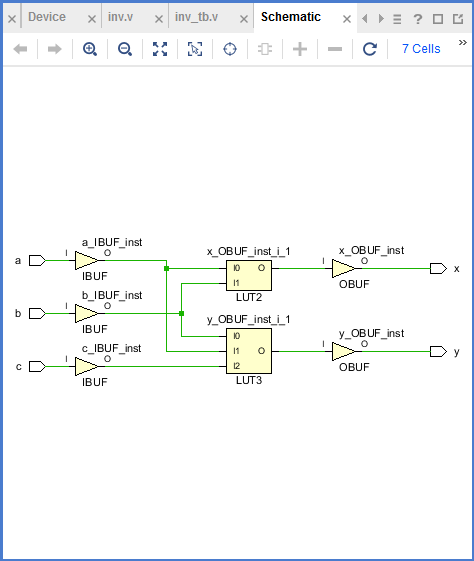
****

**A, B 둘 다 clk의 riging edge부터 aa, bb, cc 각각 100, 50, 20ns 후에 값이 반전 된다. A에서 x는 aa & bb & cc 값이다. B에서 x는 aa & bb 값이고 y는 x & cc 값이며 A에서의 x값과 같다.**

**2-4. 진리표**

|  |  |
| --- | --- |
| **(A) 1output**  C:\Users\KSH\Desktop\3in1outandtable.png | **C:\Users\KSH\Desktop\end.png(B) 2outputs**  **C:\Users\KSH\Desktop\mid.png** |

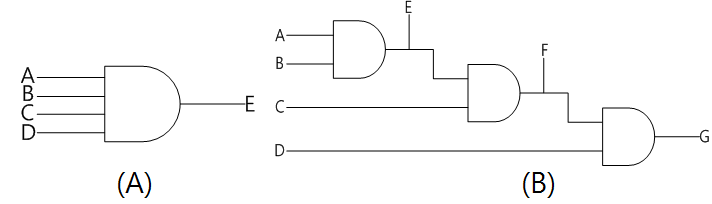
**2-5. Synthesis schematic of 2 outputs**



Simulation schematic은 처음 보였던 (A) (B) 그림과 같지만 synthesis schematic에서는 위와 같다. 차이점은 다음과 같다. 처음 그림에서는 y = x & c처럼 x를 재활용하고 여기서는 y = a & b & c 로 재활용하지 않는다..

3. 4 input AND Gate 동작 이해 및 확인

X

****

Z

Y

X

**- A: 4inputs 1output, B: 4inputs 3outputs**

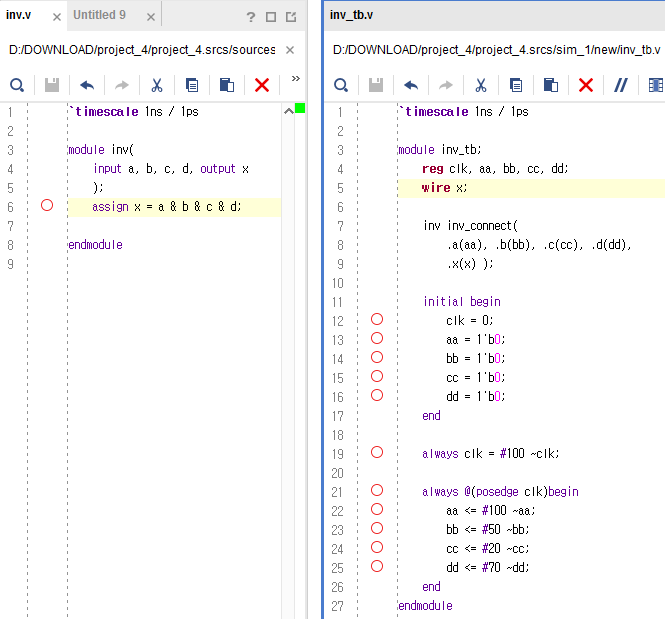
**3-1. Boolean 비교**

**(A) X = A & B & C & D**

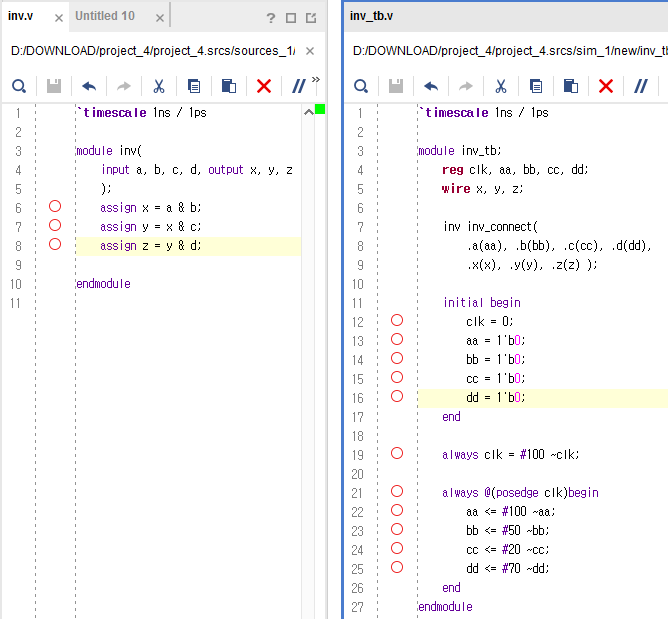
**(B) X = A & B, Y = X & C, Z = Y & D**

**3-2. 코드**

**(A) 1output**

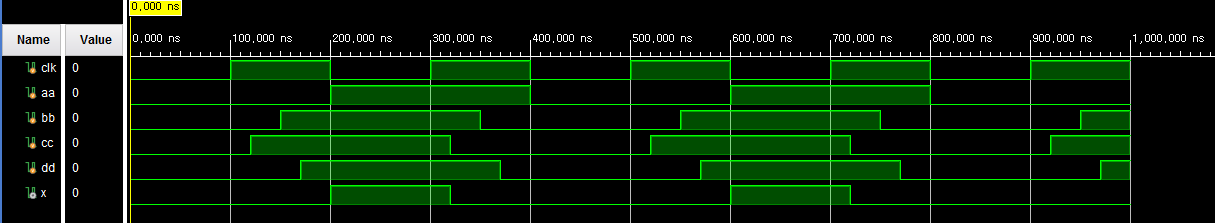
****

**(B) 3outputs**

****

**3-3. simulation**

**(A) 1output**



**(B) 3outputs**

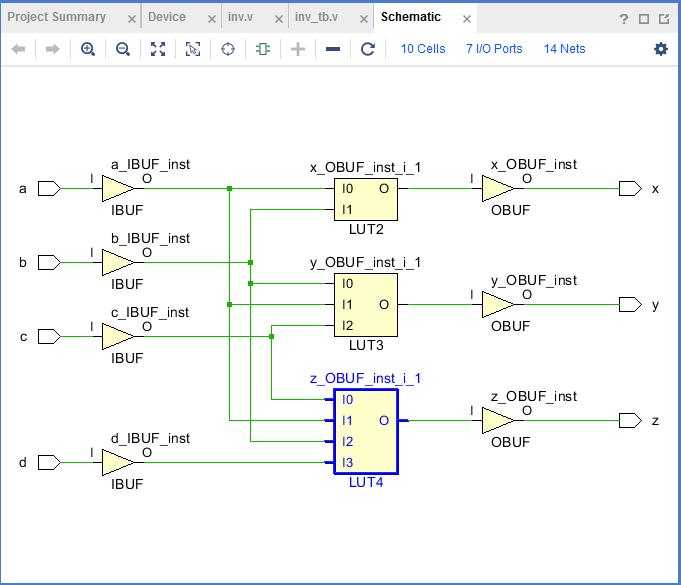
****

**A, B 둘 다 clk의 riging edge부터 aa, bb, cc, dd 각각 100, 50, 20ns, 70ns 후에 값이 반전 된다. A에서 x는 aa & bb & cc & dd 값이다. B에서 x는 aa & bb 값, y는 x & cc 값이고 z는 y & dd 값이다. B에서의 z = A에서의 x.**

**3-4. 진리표**

|  |  |
| --- | --- |
| **(A) 1output**  **C:\Users\KSH\Desktop\1p.png** | **(B) 3outputs**  **C:\Users\KSH\Desktop\33.pngC:\Users\KSH\Desktop\22.pngC:\Users\KSH\Desktop\11.png** |

**3-5. Synthesis schematic of 3 outputs**

****

3input AND gate에서처럼 synthesis schematic은 값을 재활용하지 않고 처음 input들을 그대로 넣는다. 즉 y = a & b & c, z = a & b & c & d이다.

4. 3 input OR Gate 동작 이해 및 확인

**- A: 3inputs 1output, B: 3inputs 2outputs**

**2-1. Boolean 비교**

**(A) X = A | B | C**

**(B) X = A | B, Y = X | C**

**2-2. 코드**

**(A) 1output**

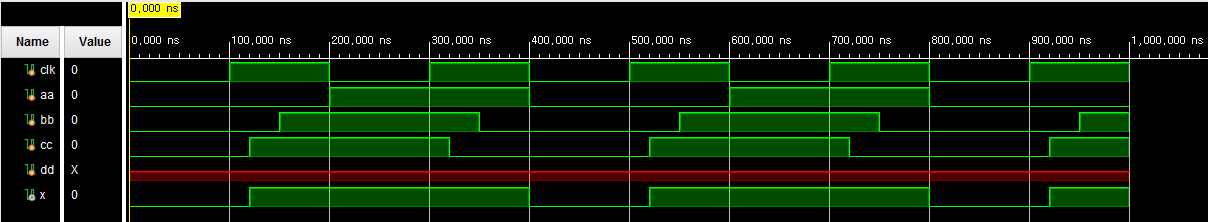
**3 input AND Gate에서의 (A) 코드와 동일 (단, ‘&’ 대신 ‘|’ 를 사용)**

**(B) 2outputs**

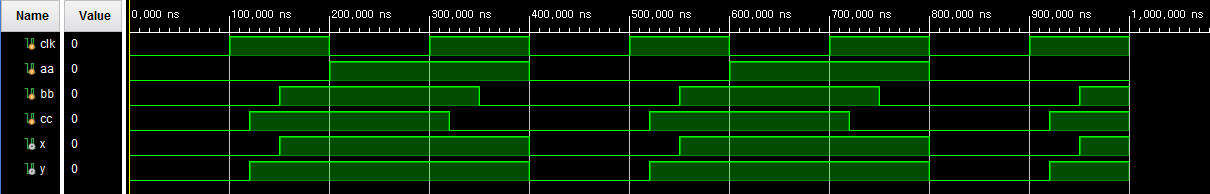
**3 input AND Gate에서의 (B) 코드와 동일 (단, ‘&’ 대신 ‘|’ 를 사용)**

**2-3. simulation**

**(A) 1output (dd는 제외)**

****

**(B) 2outputs**

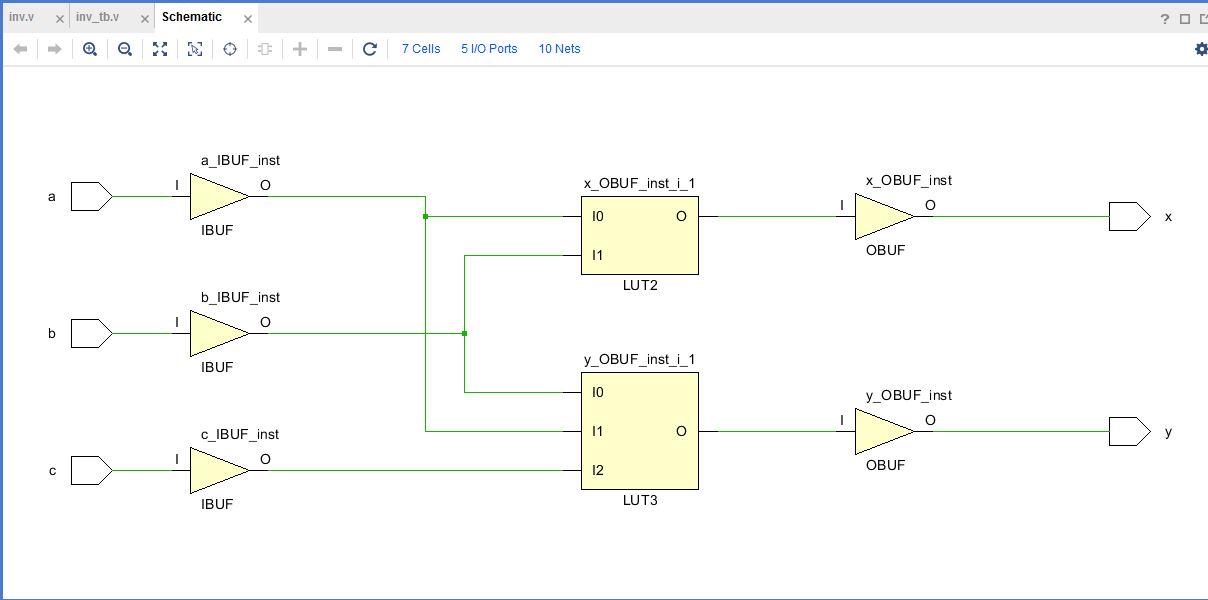
****

**A와 B의 차이는 ouput의 개수 차이일 뿐 결과는 같다. A에서 x는 aa, bb와 cc가 모두 0일 때 0이며 나머지 경우는 1이다. 이는 B에서의 y와 같다. B에서의 x는 aa와 bb가 모두 0일 때 0이다.**

**2-4. 진리표**

|  |  |
| --- | --- |
| **(A) 1output**  **C:\Users\KSH\Desktop\2.png** | **(B) 2outputs**  **C:\Users\KSH\Desktop\1.pngC:\Users\KSH\Desktop\2.png** |

**2-5. Synthesis schematic of 2 outputs**

****

**2ouputs인경우에 Synthesis schematic에서는 중간 ouput인 x값을 재활용하지 않는다.**

5. 4 input OR Gate 동작 이해 및 확인

**- A: 4inputs 1output, B: 4inputs 3outputs**

**3-1. Boolean 비교**

**(A) X = A | B | C | D**

**(B) X = A | B, Y = X | C, Z = Y | D**

**3-2. 코드**

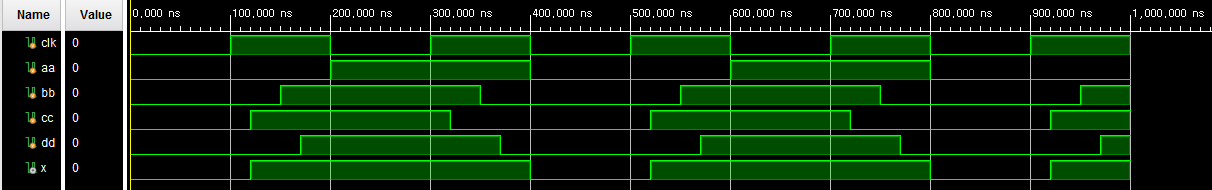
**4 input AND Gate에서의 (A) 코드와 동일 (단, ‘&’ 대신 ‘|’ 를 사용)**

**(B) 3outputs**

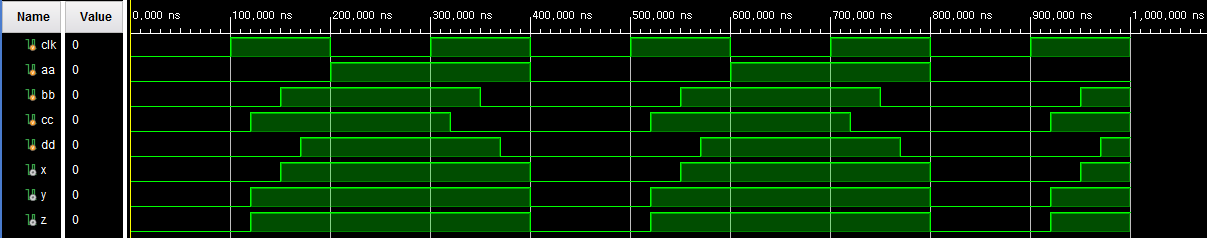
**4 input AND Gate에서의 (B) 코드와 동일 (단, ‘&’ 대신 ‘|’ 를 사용)**

**3-3. simulation**

**(A) 1output**

****

**(B) 3outputs**

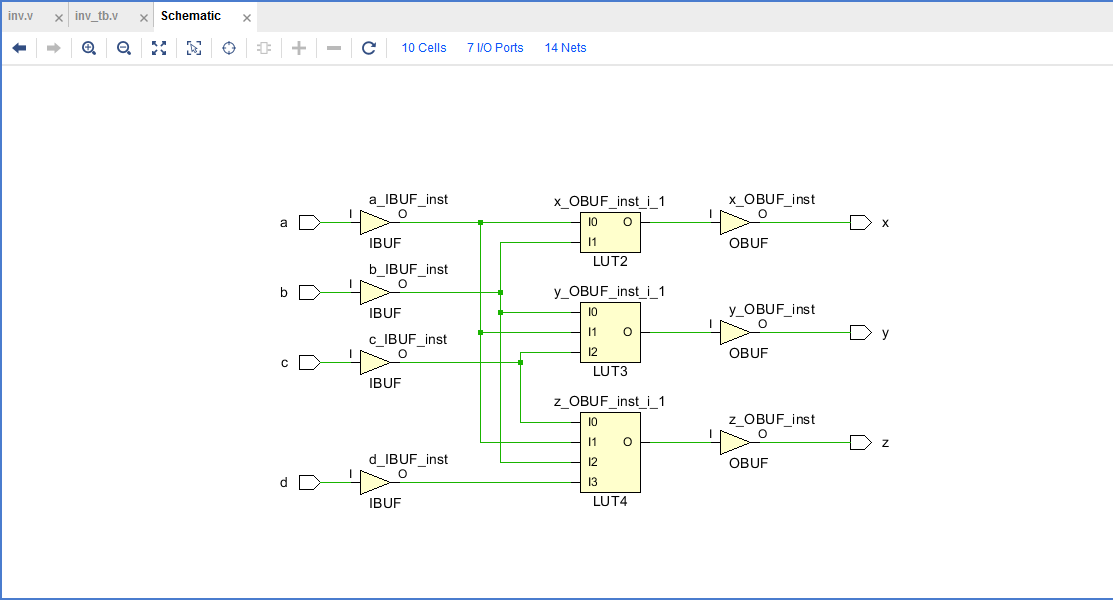
****

**A와 B의 차이는 중간 결과 두 개가 있냐 없냐의 차이만 있다. 즉 output의 개수만 다를 뿐 최종 결과 값은 같다.**

**3-4. 진리표**

|  |  |
| --- | --- |
| **(A) 1output**  **C:\Users\KSH\Desktop\3.png** | **(B) 3outputs**  **C:\Users\KSH\Desktop\3.pngC:\Users\KSH\Desktop\2.pngC:\Users\KSH\Desktop\1.png** |

**3-5. Synthesis schematic of 3 outputs**



마찬가지로 synthesis schematic에서는 재활용을 하지 않는다.6. 결과 검토

다중 input gate의 경우 기술적으로 가능한 한 개의 gate를 통해 input을 처리하는 것이 좋아 보인다. 만약 중간 결과를 재사용하는 경우가 있다면 어쩔 수 없이 여러 개의 gate를 사용해야 할 것이다.

**7. 추가 조사**

논리 게이트의 입력 개수가 늘어날수록 perforaence가 좋아질 거 같지만 일정 개수를 넘어가면 물리적인 한계로 오히려 떨어진다.