컴퓨터공학실험2 3주차 결과보고서

전공: 컴퓨터공학과 학년: 2 학번: 20191559 이름: 강상원

1. 실험 목적

AND, OR gate의 실제 수행을 확인해본다. Verilog 언어를 이용해 다중 AND, OR gate를 구현하는 방법을 익힌다. Schematic 구조, Simulation을 통해 gate의 구현을 확인한다.

경우의 수를 나누어 각 입력에 대한 gate의 출력값을 확인해 본다.

2. FPGA 동작법을 설명하시오.

Verilog로 코딩을 한 이후, 설정된 FPGA 기기에 핀을 할당한다. (작성한 Verilog 코드의 포트를 핀에 할당)

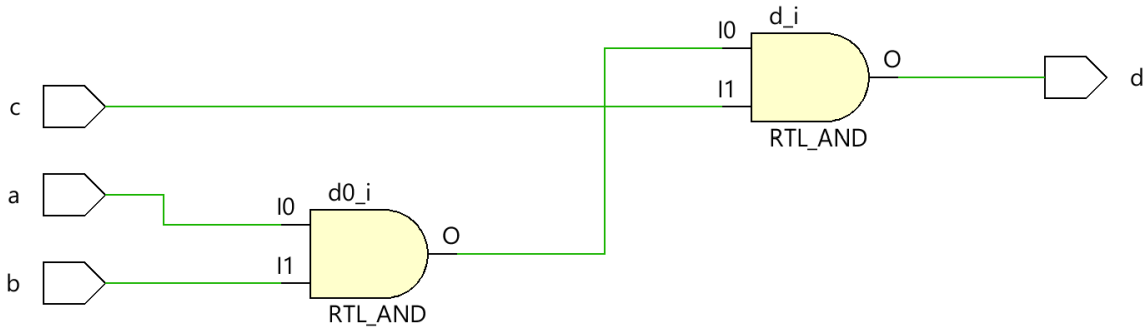
“Run Synthesis”, “Run Implementation” 이후 “Generate Bitstream”을 통해 FPGA가 정보를 넘겨받게 한다. USB로 컴퓨터에 연결된 FPGA의 동작을 확인하여 작성한 Verilog 코드를 검증한다.

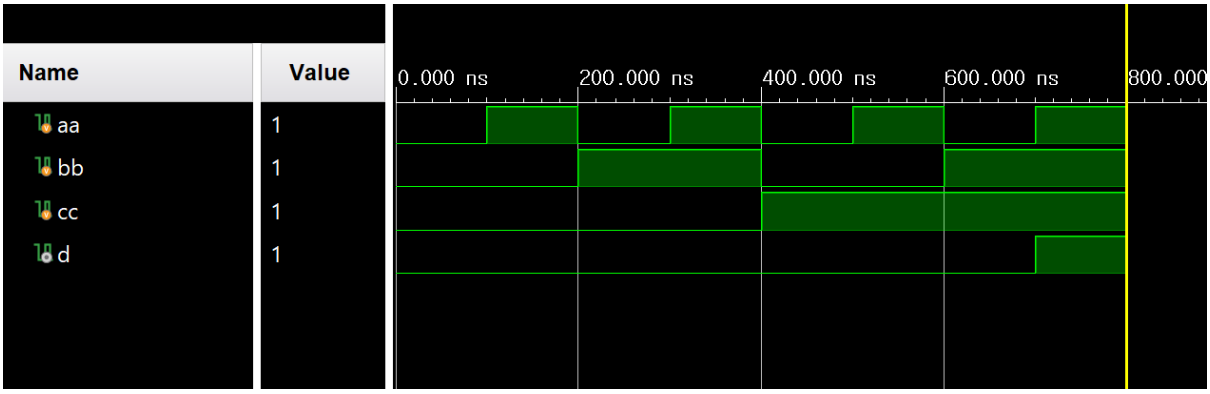
3. 3-input AND gate의 simulation 결과 및 과정에 대해서 설명하시오. (3 input, 2 output)

(A)는 assign d = a & b & c와 같이 나타낼 수 있다. Schematic과 Simulation 결과는 아래와 같다.

```
`timescale 1ns / 1ps

module three_input_and_gate_a(
    input a,
    input b,
    input c,
    output d
    );

    assign d = (a&b&c);

endmodule
```

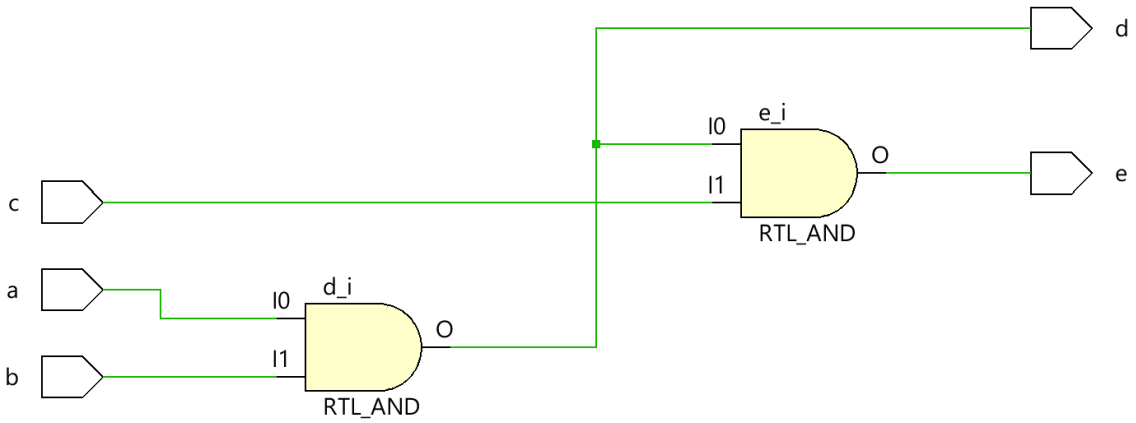
Schematic과 Simulation은 다음쪽에

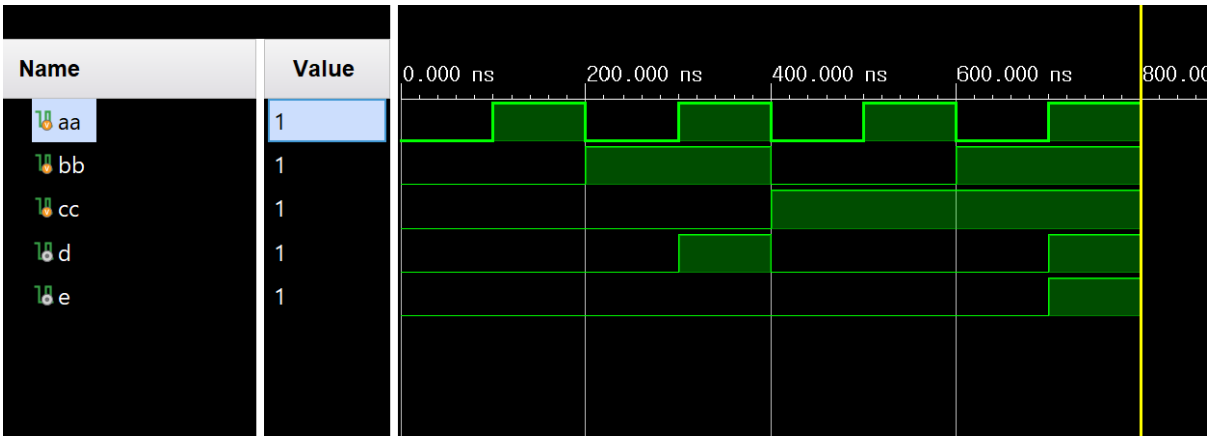




(B)는 assign d = a & b, assign e = d & c와 같이 나타낼 수 있다. Schematic과 Simulation 결과는 아래와 같다.

```
`timescale 1ns / 1ps

module three_input_and_gate_b(
    input a,
    input b,
    input c,
    output d,
    output e
    );

    assign d = a&b;
    assign e = d&c;

endmodule
```





(A)와 (B)의 시뮬레이션 결과를 비교해 보면 최종 결과값이 동일하다는 사실을 알 수 있다.

아래는 3-input AND gate의 진리표이다. (중복이므로 편의상 (B) 기준으로 통합해 작성)

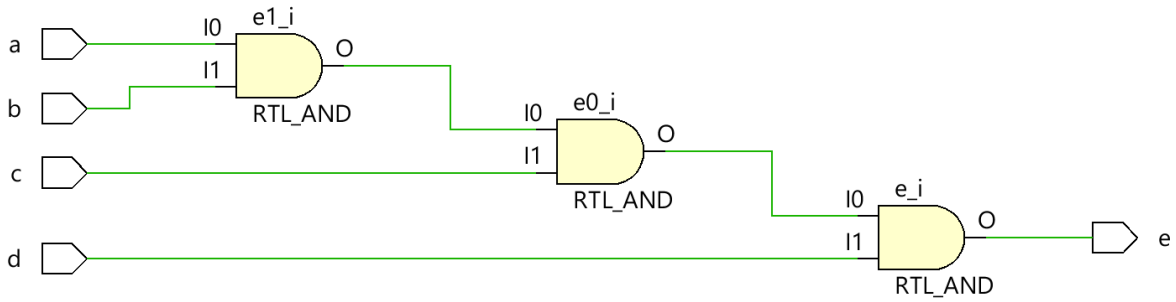
Input a	Input b	Input c	Output d	Output e
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	0	0
1	0	0	0	0
1	0	1	0	0
1	1	0	1	0
1	1	1	1	1

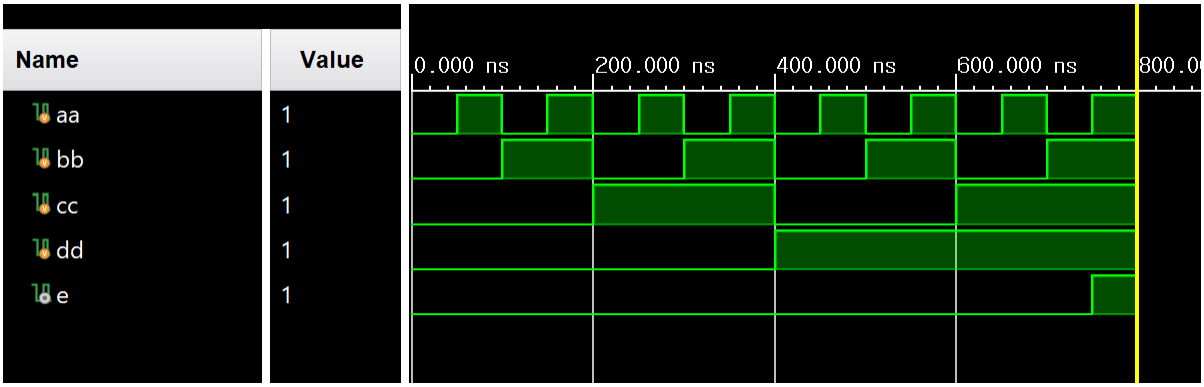
Input a, b, c의 값이 모두 1이 되어야 최종 output이 1이 됨을 확인할 수 있다.

4. 4-input AND gate의 simulation 결과 및 과정에 대해서 설명하시오. (4 input, 3 output)

(A)는 assign e = a & b & c & d와 같이 나타낼 수 있다. Schematic과 Simulation 결과는 아래와 같다.

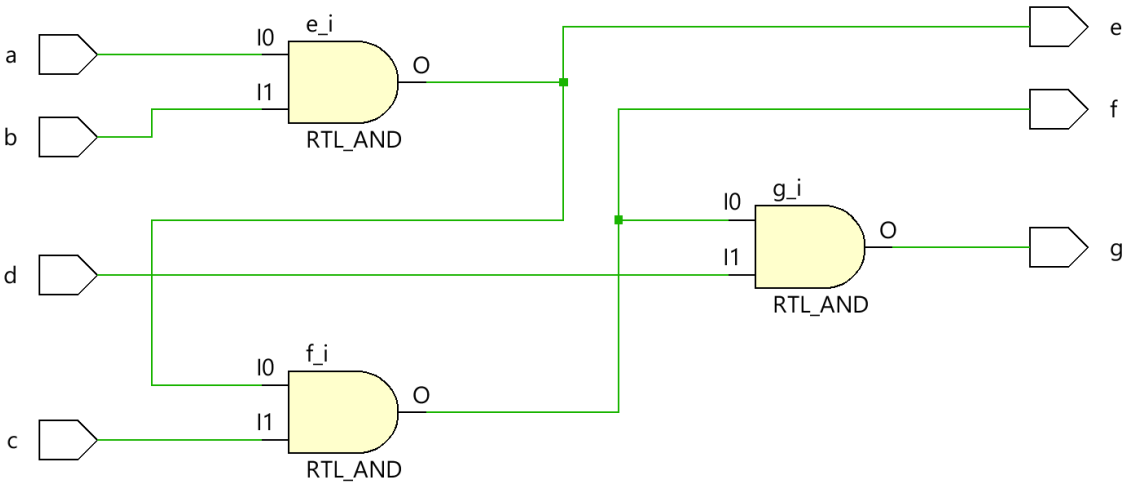
```
`timescale 1ns / 1ps

module four_input_and_gate_a(
    input a, b, c, d,
    output e
    );

    assign e = (a & b & c & d);

endmodule
```

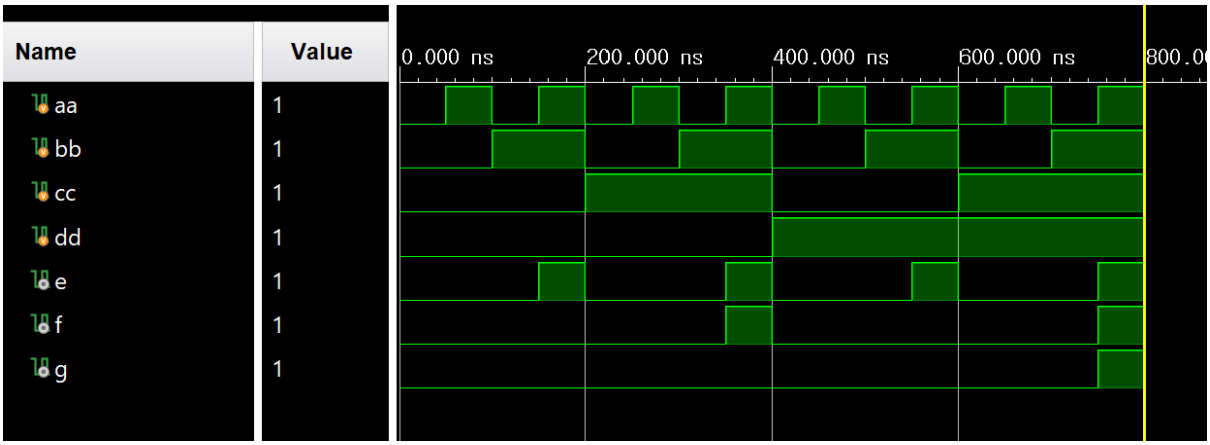




(B)는 assign e = a & b, assign f = e & c, assign g = f & d와 같이 나타낼 수 있다. Schematic과 Simulation 결과는 아래와 같다.

```
`timescale 1ns / 1ps

module four_input_and_gate_b(
    input a, b, c, d,
    output e, f, g
    );

    assign e = a & b;
    assign f = e & c;
    assign g = f & d;

endmodule
```





(A)와 (B)의 시뮬레이션 결과를 비교해 보면 최종 결과값이 동일하다는 사실을 알 수 있다.

다음은 4-input AND gate의 진리표이다. (중복이므로 편의상 (B) 기준으로 통합해 작성)

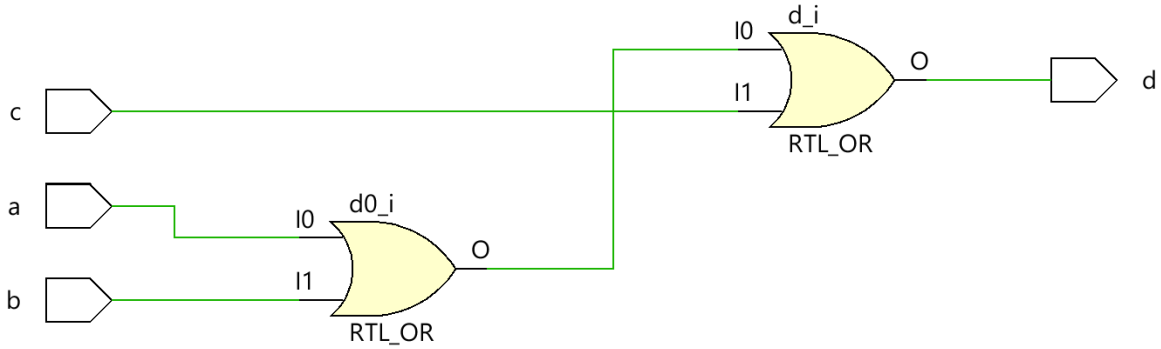
Input a	Input b	Input c	Input d	Output e	Output f	Output g
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	0	1	0	0	0	0
0	0	1	1	0	0	0
0	1	0	0	0	0	0
0	1	0	1	0	0	0
0	1	1	0	0	0	0
0	1	1	1	0	0	0
1	0	0	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	0	0	0
1	0	1	1	0	0	0
1	1	0	0	1	0	0
1	1	0	1	1	0	0
1	1	1	0	1	1	0
1	1	1	1	1	1	1

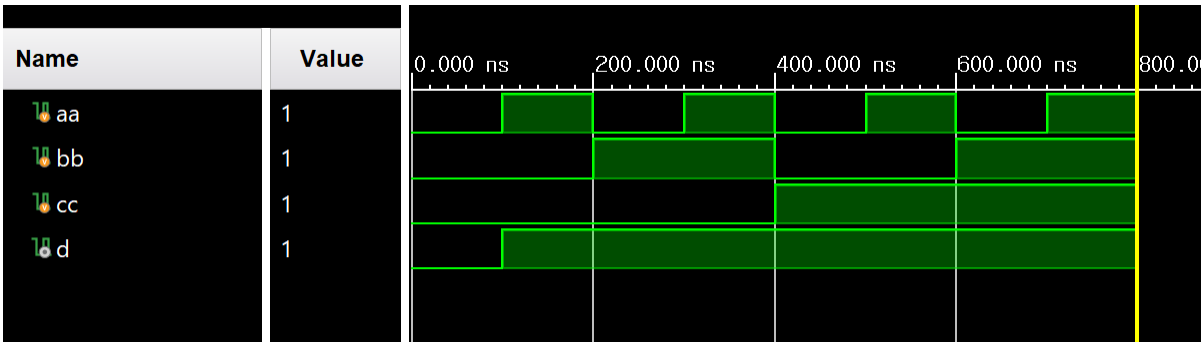
Input a, b, c, d의 값이 모두 1이 되어야 최종 output이 1이 됨을 확인할 수 있다.

5. 3-input OR gate의 simulation 결과 및 과정에 대해서 설명하시오. (3 input, 2 output)

(A)는 assign d = a | b | c와 같이 나타낼 수 있다. Schematic과 Simulation 결과는 아래와 같다.

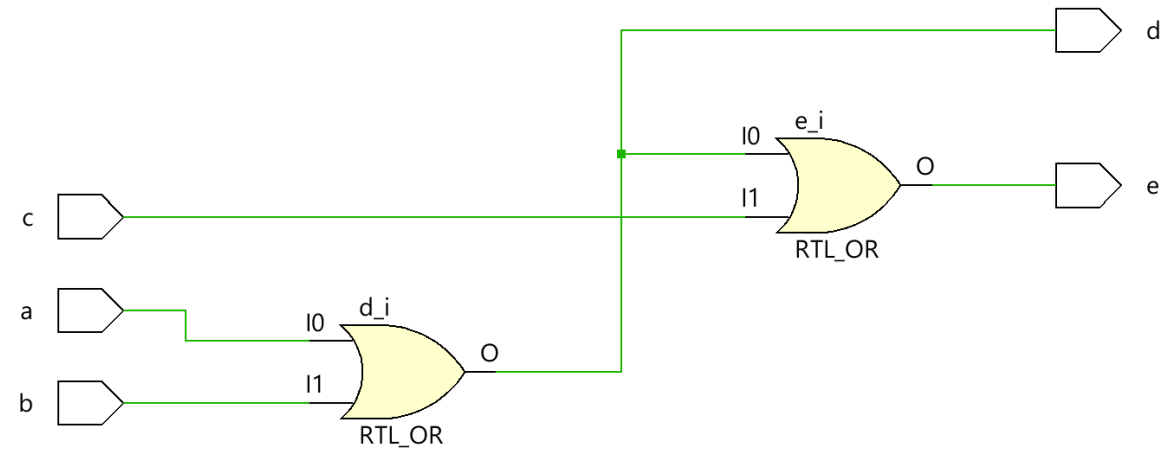
```
`timescale 1ns / 1ps

module three_input_or_gate_a(
    input a, b, c,
    output d
    );

    assign d = (a | b | c);

endmodule
```

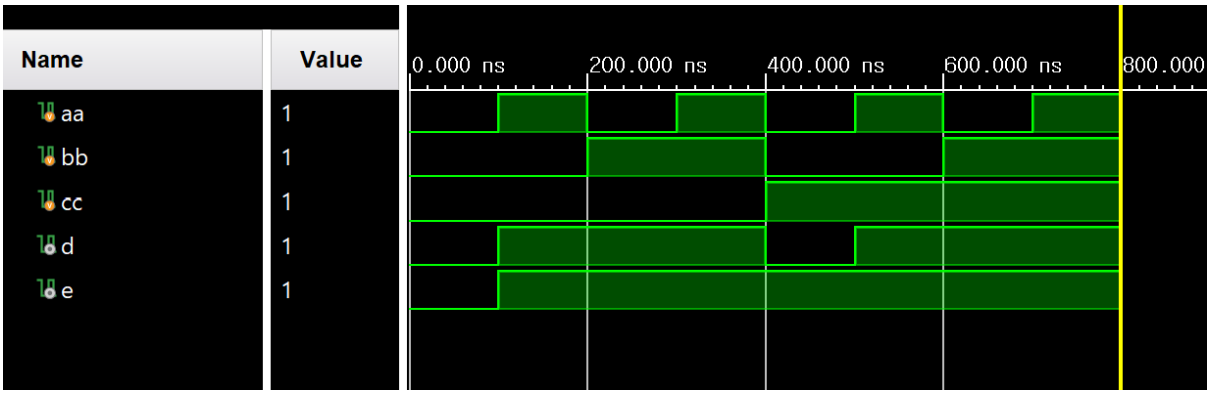




(B)는 assign d = a | b, assign e = d | c와 같이 나타낼 수 있다. Schematic과 Simulation 결과는 아래와 같다.

```
`timescale 1ns / 1ps

module three_input_or_gate_b(
    input a, b, c,
    output d, e
    );

    assign d = a | b;
    assign e = d | c;

endmodule
```





(A)와 (B)의 시뮬레이션 결과를 비교해 보면 최종 결과값이 동일하다는 사실을 알 수 있다.
다음은 3-input OR gate의 진리표이다. (중복이므로 편의상 (B) 기준으로 통합해 작성)

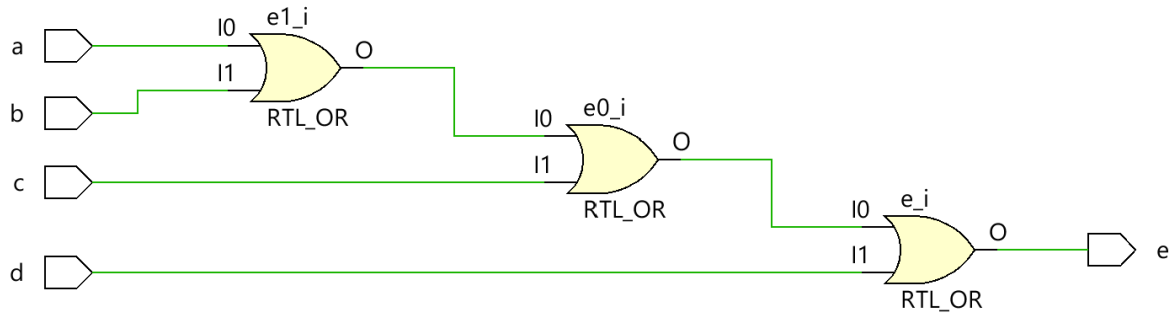
Input a	Input b	Input c	Output d	Output e
0	0	0	0	0
0	0	1	0	1
0	1	0	1	1
0	1	1	1	1
1	0	0	1	1
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

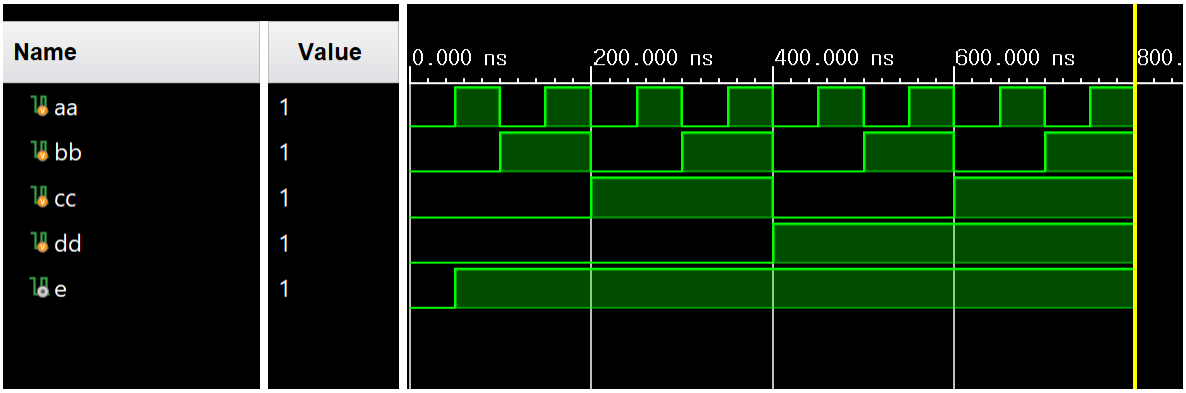
Input a, b, c 중 하나라도 1이면 최종 output이 1이 됨을 확인할 수 있다.

6. 4-input OR gate의 simulation 결과 및 과정에 대해서 설명하시오. (4 input, 3 output)

(A)는 assign e = a | b | c | d와 같이 나타낼 수 있다. Schematic과 Simulation 결과는 아래와 같다.

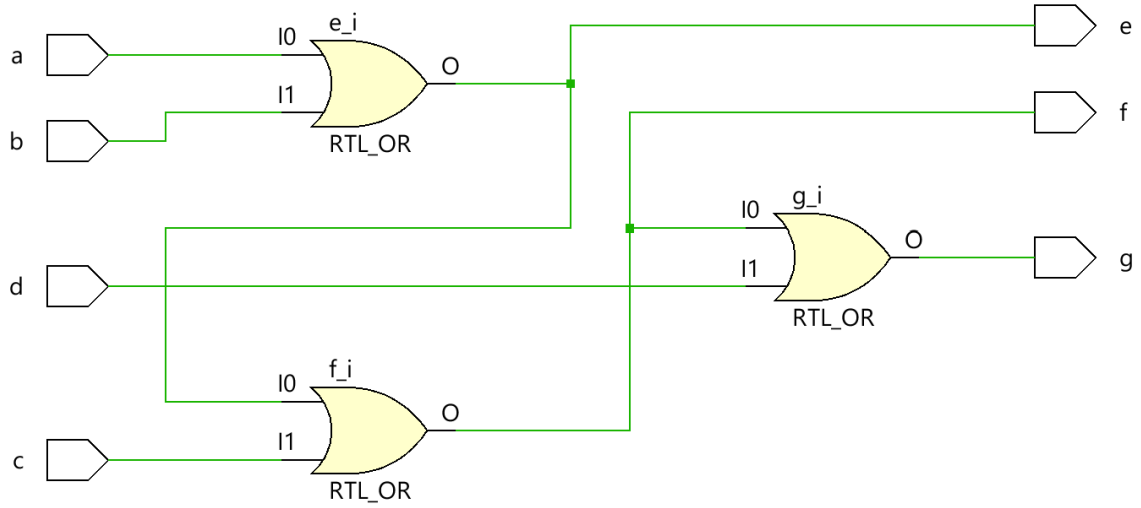
```
`timescale 1ns / 1ps

module four_input_or_gate_a(
    input a, b, c, d,
    output e
    );

    assign e = (a | b | c | d);

endmodule
```

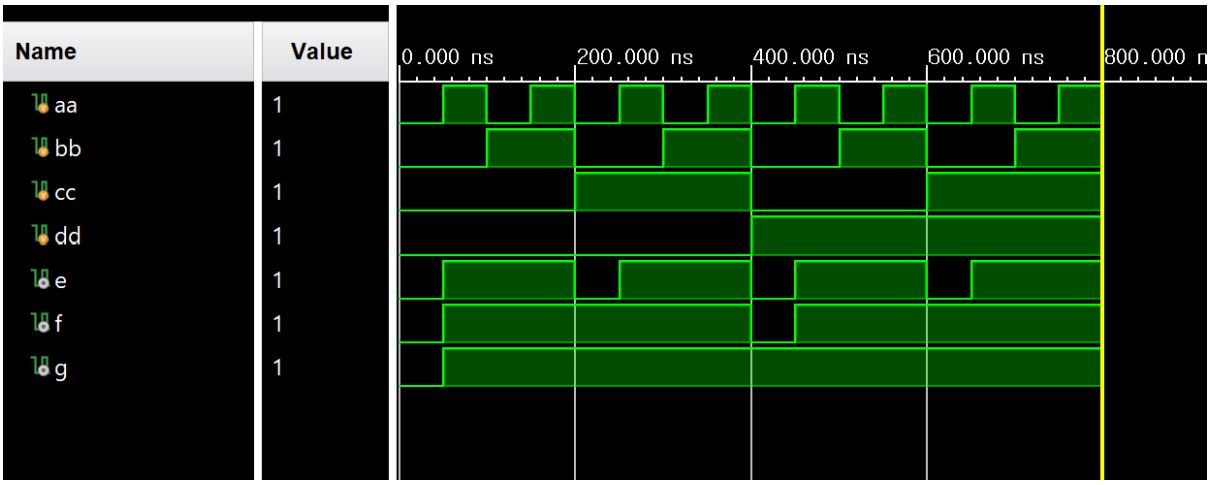




(B)는 assign e = a | b, assign f = e | c, assign g = f | d와 같이 나타낼 수 있다. Schematic과 Simulation 결과는 아래와 같다.

```
`timescale 1ns / 1ps

module four_input_or_gate_b(
    input a, b, c, d,
    output e, f, g
    );

    assign e = a | b;
    assign f = e | c;
    assign g = f | d;

endmodule
```





(A)와 (B)의 시뮬레이션 결과를 비교해 보면 최종 결과값이 동일하다는 사실을 알 수 있다.

다음은 4-input OR gate의 진리표이다. (중복이므로 편의상 (B) 기준으로 통합해 작성)

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	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

Input a, b, c, d 중 하나라도 1이면 최종 output이 1이 됨을 확인할 수 있다.

7. 결과 검토 및 논의사항.

3, 4-input AND, OR gate를 Verilog 코딩하여 시뮬레이션과 Schematic을 확인해 보았다. 이 때 a & b & c와 같은 수식은 Schematic 상에서 PPT의 (A) 모양이 나오지 않는데, 이는 Verilog에서 기본적으로 3개를 한꺼번에 연산하는 것으로 나타내지 않고 따로 풀어서 나타내기 때문이다.

결론적으로 시뮬레이션을 비교한 결과 (A)와 (B)는 모두 서로 같은 값이 나왔다. 이는 Schematic 구조가 조금씩 다르거나, 수식이 차이가 나더라도 같은 출력을 가지는 회로를 구성할 수 있다는 것을 뜻한다.

∴ 효율적인 방향으로 Verilog 코드를 작성해 회로를 설계해야 한다.

3, 4-input AND gate는 모든 입력 값이 1(High)이어야지만 출력이 1이 되고,

3, 4-input OR gate는 입력값 중 하나라도 1이면 출력이 1이 됨을 확인할 수 있었다.

8. 추가 이론 조사 및 작성.

부울 대수(Boolean Algebra)에서 자주 쓰이는 정리를 정리해 보았다.

기본적으로 **교환·결합·분배 법칙**이 적용된다.

교환법칙: $a + b = b + a, a \cdot b = b \cdot a$

결합법칙: $(a \cdot b) \cdot c = a \cdot (b \cdot c)$

분배법칙: $a \cdot (b + c) = a \cdot b + ac$

(여기서 $\cdot$, +는 각각 논리곱, 논리합을 의미)

이외에도 드모르간의 법칙, 흡수 법칙 등이 존재한다.

드모르간의 법칙: $(a + b)' = a' \cdot b', (a \cdot b)' = a' + b'$

흡수 법칙: $a + a \cdot b = a \cdot (1 + b) = a \cdot 1 = a$ … 이 이외의 법칙들(연산 방법들)은 위에서 소개한 정리들로 모두 추론 가능하다.

AND, OR gate 이외에도 **NAND, NOR, XOR gate** 등이 존재한다.

NAND gate는 AND gate의 결과값에 NOT을 취한 것과 같으며, 진리표는 다음과 같다.

Input a	Input b	Output c
0	0	1
0	1	1
1	0	1
1	1	0

NOR gate는 OR gate의 결과값에 NOT을 취한 것과 같으며, 진리표는 다음과 같다.

Input a	Input b	Output c
0	0	1
0	1	1
1	0	1
1	1	0

XOR gate는 입력값 중에 1(High)이 홀수개일 때 1을 출력한다. 진리표는 다음과 같다.

Input a	Input b	Output c
0	0	1
0	1	1
1	0	1
1	1	0