디지털시스템설계 실습 보고서

실험 1. 논리회로 기초

무은재학부

20210716 최대현

1. 연산의 진리표 작성

신호 A, B에 대한 NOT, OR, AND, NOR, NAND의 연산 결과를 진리표로 작성하면 다음과 같다.

<NOT>

A	A'
0	1
1	0

<OR>

A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1

<AND>

A	B	A·B
0	0	0
0	1	0
1	0	0
1	1	1

<NOR>

A	B	$\overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0

<NAND>

A	B	$\overline{A \cdot B}$
0	0	1
0	1	1
1	0	1
1	1	0

2. 논리 게이트의 Verilog 구현

2-1) lab1_1: AND 게이트의 Verilog 구현

AND 게이트를 구현한 source 파일은 다음과 같다. 입력 와이어 inA와 inB, 출력 와이어 outAND를 만들어준 뒤 and gate modeling으로 연결해준 코드이다.

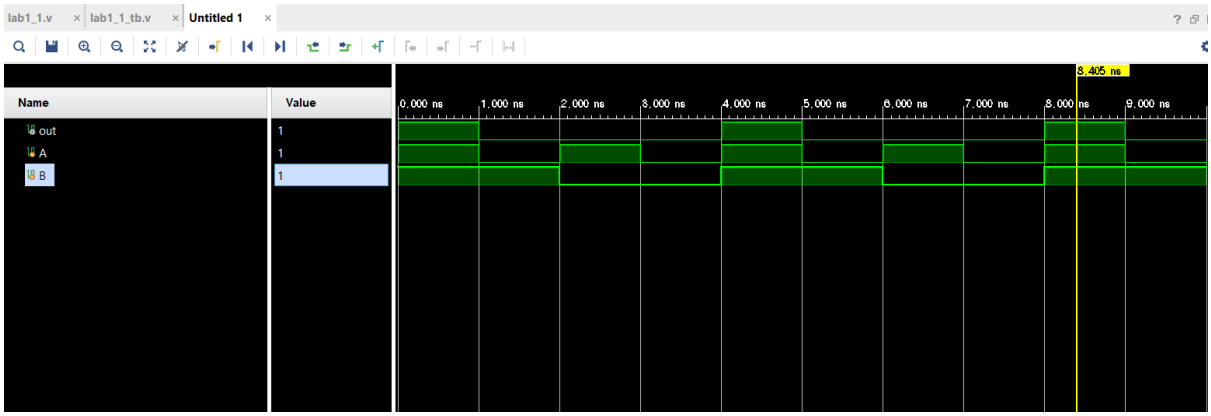
```
1  /* CSED273 lab1 experiment 1 */
2  /* lab1_1.v */
3
4
5  /* Implement AND */
6  module lab1_1(outAND, inA, inB);
7      output wire outAND;
8      input wire inA, inB;
9
10     ////////////////////////
11     /* Add your code here */
12     and(outAND, inA, inB);
13     ////////////////////////
14
15 endmodule
```

그리고 이 파일을 시뮬레이션하기 위해 만든 testbench code는 다음과 같다. Module lab1_1을 그대로 가져와서 AND라는 이름으로 선언해줬고, 입력값 A와 B의 초기값은 1로 설정한 뒤 10번의 timescale 이후 종료시켰다. 이 때, A는 1 timescale마다 반전시키고 B는 2 timescale마다 반전시켰다.

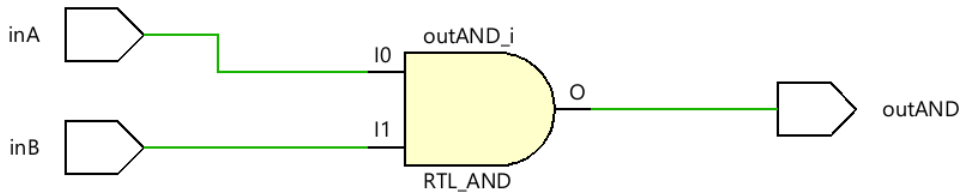
C:/Users/user/Desktop/lab/lab1/lab1_1_tb.v

```
1  /* CSED273 lab1 experiment 1 */
2  /* lab1_1_tb.v */
3
4  `timescale 1ns / 1ps
5  module lab1_1_tb();
6      wire out;
7      reg A, B;
8      lab1_1 AND(out, A,B);
9      /* Initialize A and B */
10     initial begin
11         ////////////////////////
12         A=1;
13         B=1;
14         #10 $finish;
15         ////////////////////////
16     end
17     /* Flip A every 1ns */
18     always begin
19         #1 A <= !A;
20     end
21     /* Flip B every 2ns */
22     always begin
23         ////////////////////////
24          #2 B <= !B;
25         ////////////////////////
26     end
27 endmodule
28
```

이를 10ns동안 시뮬레이션해본 결과는 다음과 같다.



또한, schematic 기능을 이용해 생성한 회로도는 다음과 같다.

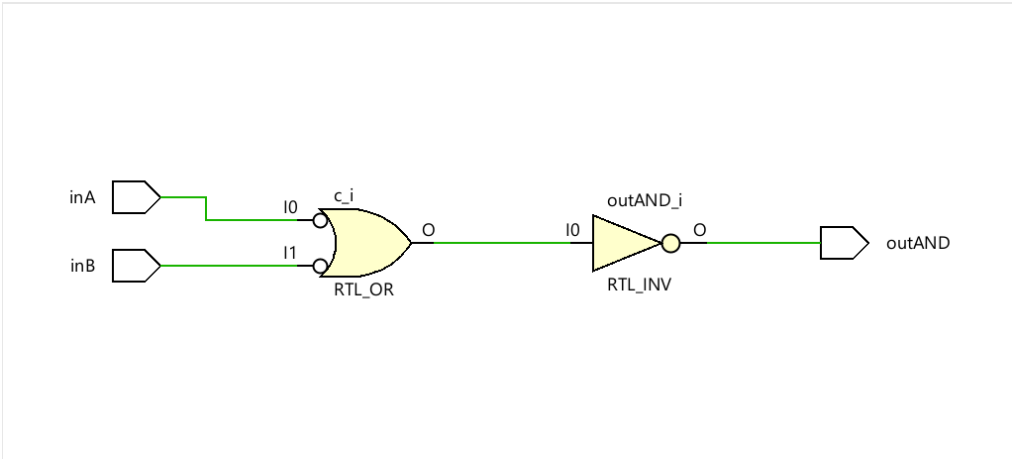


2-2) lab1-2_1: {OR, NOT} functionally complete의 Verilog 구현

OR와 NOT 게이트만으로 AND 게이트를 구현한 source 파일은 다음과 같다. 별도의 와이어 na, nb, c를 만들었다. na와 nb는 A, B를 not gate로 반전시킨 값을 전달하고, c는 na와 nb를 or 연산한 값을 전달한다. 마지막으로 이것을 not 연산 해주면 OR, NOT 연산만 가지고 AND 연산을 만들어낼 수 있다. 즉 {OR, NOT} 집합은 functionally complete하다.

```
4
5  /* Implement AND with {OR, NOT}
6   * You are only allowed to wire modules together */
7  module lab1_2_1(outAND, inA, inB);
8      output wire outAND;
9      input wire inA, inB;
10
11     ////////////////////////
12     /* Add your code here */
13     wire nA;
14     wire nB;
15     wire c;
16     not(nA, inA);
17     not(nB, inB);
18     or(c, nA, nB);
19     not(outAND, c);
```

또한, schematic 기능을 이용해 생성한 회로도는 다음과 같다.



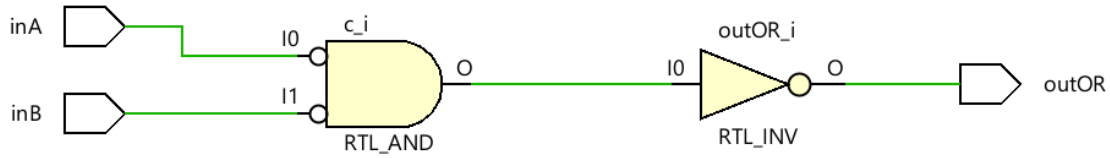
2-3) lab1-2_2: {AND, NOT} functionally complete의 Verilog 구현

AND와 NOT 게이트만으로 OR게이트를 구현한 source 파일은 다음과 같다.

별도의 wire na, nb, c를 구현했다. na, nb는 각각 inA, inB를 not gate로 반전시키는 역할이다. C는 na, nb를 and 연산한 결과를 전달하는 역할이고 이것을 not gate로 반전시키면 and와 not 연산만 가지고 or 연산을 만들어낼 수 있다. 즉, {AND, NOT} 집합은 functionally complete하다.

```
1  /* CSED273 lab1 experiment 2_ii */
2  /* lab1_2_ii.v */
3
4
5  /* Implement OR with {AND, NOT}
6   * You are only allowed to wire modules together */
7  module lab1_2_ii(outOR, inA, inB);
8      output wire outOR;
9      input wire inA, inB;
10     wire nA, nB;
11     wire c;
12     ////////////////////////
13     /* Add your code here */
14     not(nA,inA);
15     not(nB, inB);
16     and(c, nA, nB);
17     not(outOR, c);
18     ////////////////////////
19
20 endmodule
```

또한, schematic 기능을 이용해 생성한 회로도는 다음과 같다.



2-4) lab1-2_3: {NAND} functionally complete의 Verilog 구현

AND와 NOT 게이트만으로 OR게이트를 구현한 source 파일은 다음과 같다.

```
1  /* CSED273 lab1 experiment 2_iii */
2  /* lab1_2_iii.v */
3
4
5  module lab1_2_iii(outAND, outOR, outNOT, inA, inB);
6      output wire outAND, outOR, outNOT;
7      input wire inA, inB;
8
9      AND(outAND, inA, inB);
10     OR(outOR, inA, inB);
11     NOT(outNOT, inA);
12
13 endmodule
```

```
14
15
16  /* Implement AND, OR, and NOT modules with {NAND}
17   * You are only allowed to wire modules below
18   * i.e.) No and, or, not, etc. Only nand, AND, OR, NOT are available*/
19  module AND(outAND, inA, inB);
20      output wire outAND;
21      input wire inA, inB;
22
23      ////////////////////////
24      /* Add your code here */
25      wire anb;
26      nand(anb, inA, inB);
27      nand(outAND, anb, anb);
28
29      ////////////////////////
30
31  endmodule
```

NAND gate만으로 AND를 구현할 경우 A, B를 NAND 연산하고, (anb) 이것을 자기 자신과 NAND 연산해줌으로서 만들어낼 수 있다.

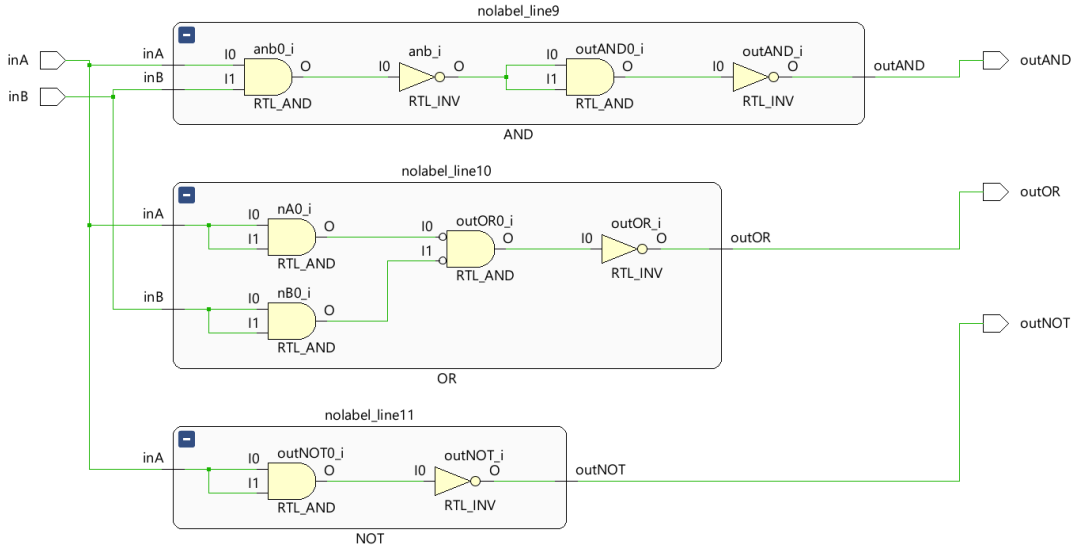
```
32
33  module OR(outOR, inA, inB);
34      output wire outOR;
35      input wire inA, inB;
36
37      ////////////////////////
38      /* Add your code here */
39      wire nA;
40      wire nB;
41      nand(nA, inA, inA);
42      nand(nB, inB, inB);
43      nand(outOR, nA, nB);
44      ////////////////////////
45
46  endmodule
47
48  module NOT(outNOT, inA);
49      output wire outNOT;
50      input wire inA;
51
52      ////////////////////////
53      /* Add your code here */
54      nand(outNOT, inA, inA);
55      ////////////////////////
56
57  endmodule
```

NAND gate만으로 OR을 구현할 경우 A, B를 각각 자기 자신과 NAND 연산하여 A′, B′를 만들고, 이것을 NAND 연산해줌으로서 만들어낼 수 있다.

NAND gate만으로 NOT을 구현할 경우 A를 자기 자신과 NAND 연산하여 만들어낼 수 있다.

NAND 연산만으로 AND, OR, NOT을 전부 구현할 수 있으므로, {NAND} 집합은 functionally complete하다.

Schematic 기능으로 생성한 회로도는 다음과 같다.



2-5) lab1-2_4: {NOR} functionally complete의 Verilog 구현

NOR 게이트만으로 AND, OR NOT게이트를 구현한 source 파일은 다음과 같다.

```
1  /* CSED273 lab1 experiment 2_iv */
2  /* lab1_2_iv.v */
3
4
5  module lab1_2_iv(outAND, outOR, outNOT, inA, inB);
6      output wire outAND, outOR, outNOT;
7      input wire inA, inB;
8
9      AND(outAND, inA, inB);
10     OR(outOR, inA, inB);
11     NOT(outNOT, inA);
12
13 endmodule
14
15
16 /* Implement AND, OR, and NOT modules with {NOR}
17  * You are only allowed to wire modules below
18  * i.e.) No and, or, not, etc. Only nor, AND, OR, NOT are available*/
19 module AND(outAND, inA, inB);
20     output wire outAND;
21     input wire inA, inB;
22     wire nA, nB;
23     ////////////////////////
24     /* Add your code here */
25     nor(nA, inA, inA);
26     nor(nB, inB, inB);
27     nor(outAND, nA, nB);
28     ////////////////////////
```

NOR gate만으로 AND를 구현할 경우 A, B 각각 자기 자신과의 NOR 연산을 통해 A', B'을 만들어내고. 이것을 NOR 연산해줌으로서 만들어낼 수 있다.

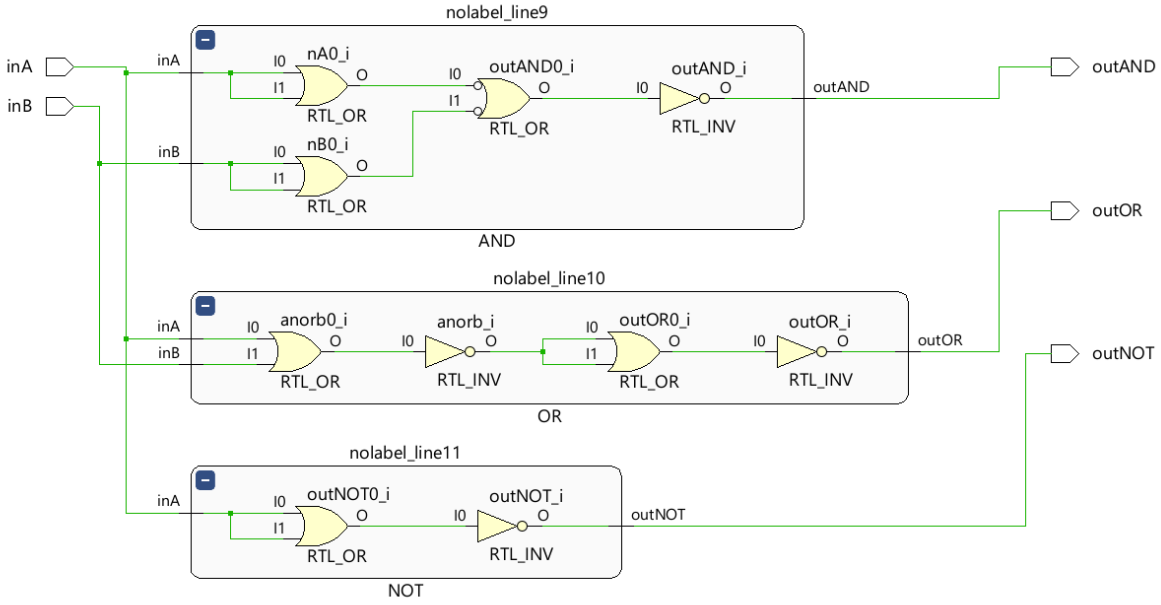
```
29
30  endmodule
31
32  module OR(outOR, inA, inB);
33      output wire outOR;
34      input wire inA, inB;
35      wire anorb;
36      ////////////////////////
37      /* Add your code here */
38      nor(anorb, inA, inB);
39      nor(outOR, anorb, anorb);
40      ////////////////////////
41
42  endmodule
43
44  module NOT(outNOT, inA);
45      output wire outNOT;
46      input wire inA;
47
48      ////////////////////////
49      /* Add your code here */
50      nor(outNOT, inA, inA);
51      ////////////////////////
52
53  endmodule
```

NOR gate만으로 OR를 구현할 경우 A, B의 NOR 연산을 한 값(anorb)을 자기 자신과 NOR 연산해줌으로서 만들어낼 수 있다.

NOR gate만으로 NOT을 구현할 경우 자기 자신과의 NOR 연산을 통해 만들어낼 수 있다.

NOR 연산만으로 AND. OR, NOT을 전부 구현할 수 있으므로, {NOR} 집합은 functionally complete하다.

Schematic 기능으로 생성한 회로도는 다음과 같다.



3. 결론

- 이번 실습을 통해 Verilog의 기초 문법을 익히고, gate-level modeling하는 방법을 숙지했다. Schematic 기능을 이용해 회로도를 생성하고, testbench 기능을 이용해 가상의 신호를 주며 simulation해볼 수 있었다. 또한, {AND, NOT}, {OR, NOT}, {NAND}, {NOR} 집합이 각각 functionally complete하다는 것을 증명할 수 있었다.