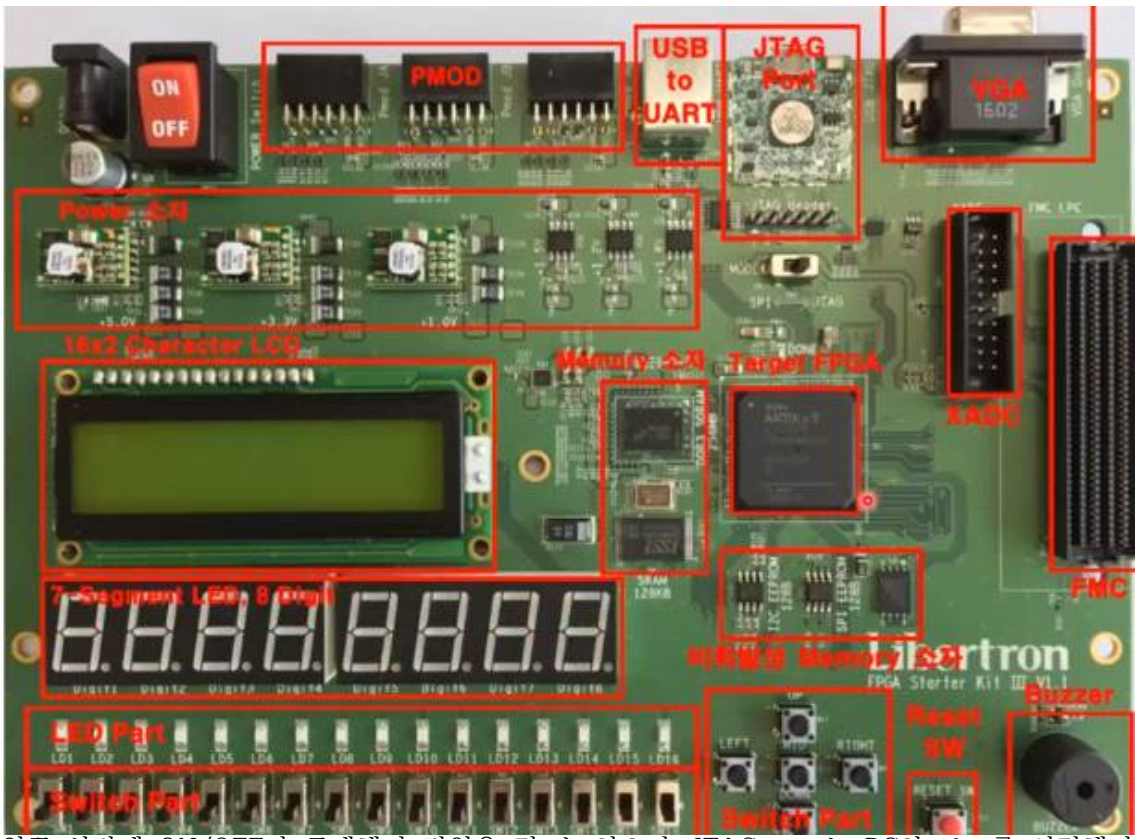
3주차 결과 보고서

20191621 이민영

1. 실험 목적

AND/OR/NOT Gate의 동작을 이해하고 확인하며, Verilog를 이용해서 다중입력 AND/OR/NOT Gate를 구현해본다. 입력 신호 생성 후 Simulation을 통해서 구현된 각각의 Gate 동작을 확인하고, FPGA를 통해서 Verilog로 구현된 회로의 동작을 확인한다.

2. FPGA 동작법을 설명하시오.

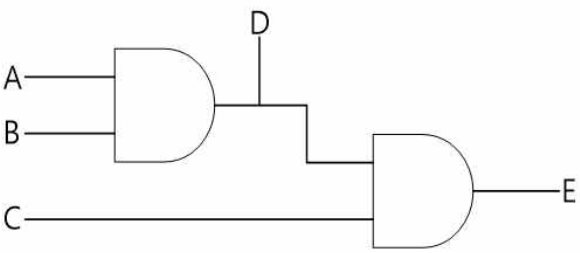


왼쪽 상단에 ON/OFF가 존재해서 전원을 킬 수 있으며, JTAG port는 PC와 보드를 연결해서 사용하기 위해서 이용한다. VGA의 경우는 화면을 연결하기 위해서 사용한다. Switch Part 부분을 이용해서 input을 조절할 수 있고, LED part는 불이 들어올 수 있는 부분이며 Output이 표시된다. Digit부분에서는 숫자를 표현할 수 있다.

Verilog를 이용해서 코딩을 한 후 device assignment를 한 후에 FPGA pin list에서 pin과 port를 링크한다. Synthesis를 시행하고 Run Implementation을 거친다. 이 후에 FPGA 보드를 보면서 결과를 확인할 수 있다.

3. 3-input AND gate의 simulation 결과 및 과정에 대해서 설명하시오.

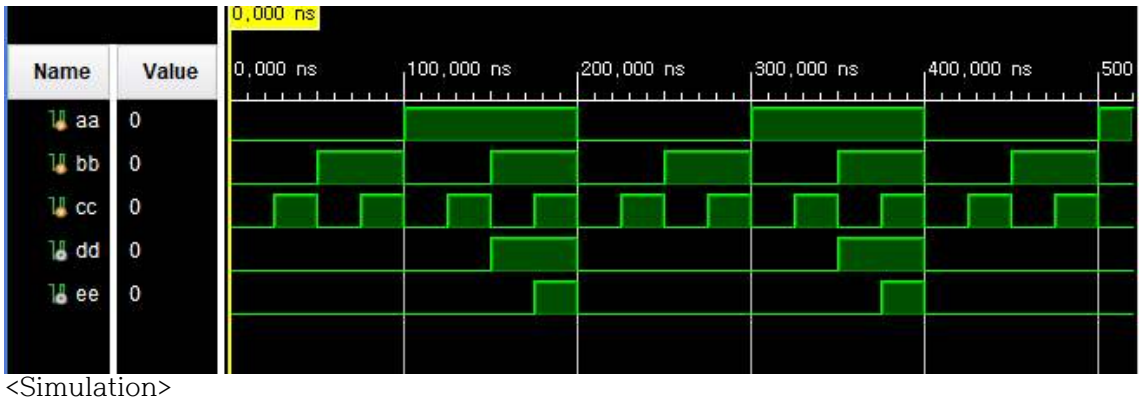
(3 input, 2 output)[3장 ppt 31 page 참조, 진리표 작성]



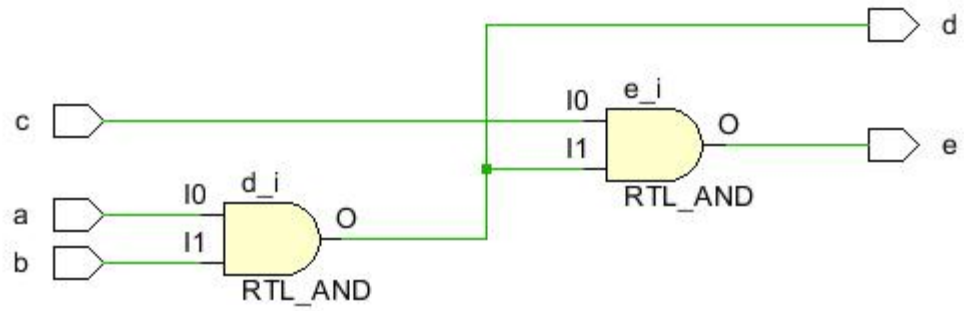
<3 input - 2 output>

3_20191621_three_input_and_b.v	3_20191621_three_input_and_b_tb.v

`timescale 1ns / 1ps
 	
module
3_20191621_three_input_and_gate_b(
input a,b,c,
output d,e
);
 	
assign d = a&b;
assign e = c&d;
endmodule
	`timescale 1ns / 1ps
	module
	3_20191621_three_input_and_gate_b_tb;
	
	reg aa;
	reg bb;
	reg cc;
	wire dd;
	wire ee;
	3_20191621_three_input_and_gate_b
	u_2_20191621_three_input_and_gate_b(
	.a (aa),
	.b (bb),
	.c (cc),
	.d (dd),
	.e (ee));
	initial aa = 1'b0;
	initial bb = 1'b0;
	initial cc = 1'b0;
	always aa = #100 ~aa;
	always bb = #50 ~bb;
	always cc = #25 ~cc;
	initial begin
	#1000
	$finish;
	end
	endmodule



<Simulation>



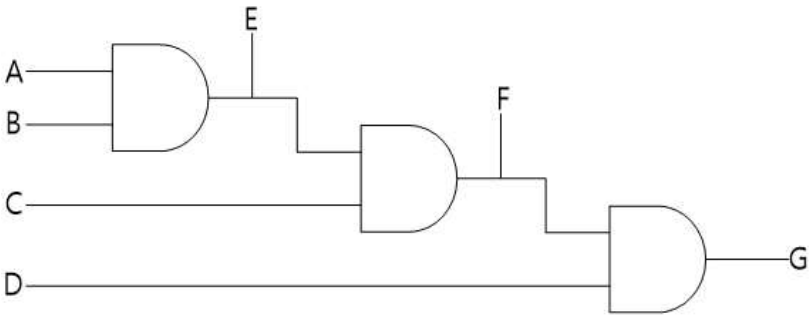
d의 값은 a & b 값으로 출력되고, e는 c&d의 값으로 출력되어서 나타나는 것을 볼 수 있다. 결과적으로 e는 a,b,c,d가 모두 1일 때만 1로 출력되는 것을 simulation을 통해서 알 수 있다.

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

<진리표>

4. 4-input AND gate의 simulation 결과 및 과정에 대해서 설명하시오.

(4 input, 3 output)[3장 ppt 33 page 참조, 진리표 작성]

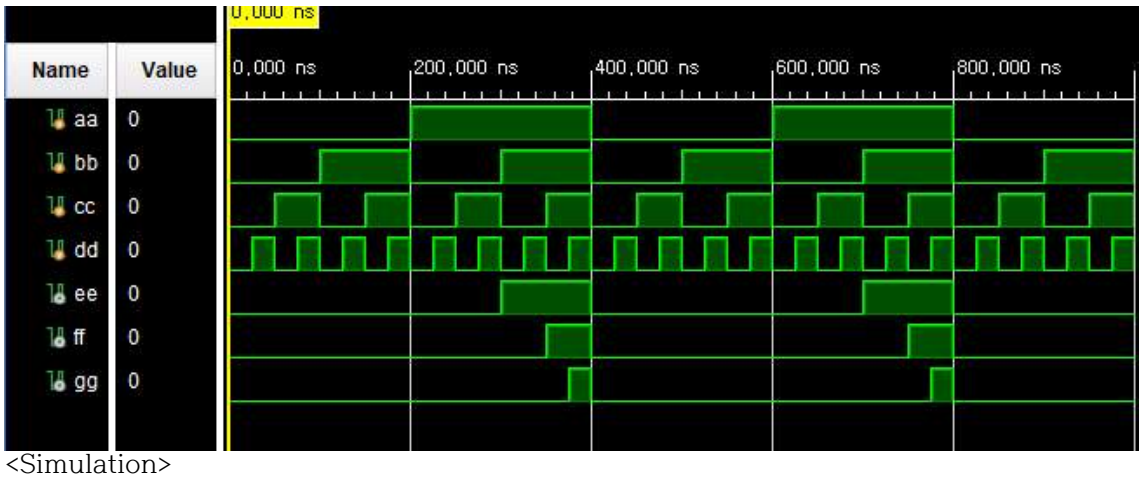


<4 input - 3 output>

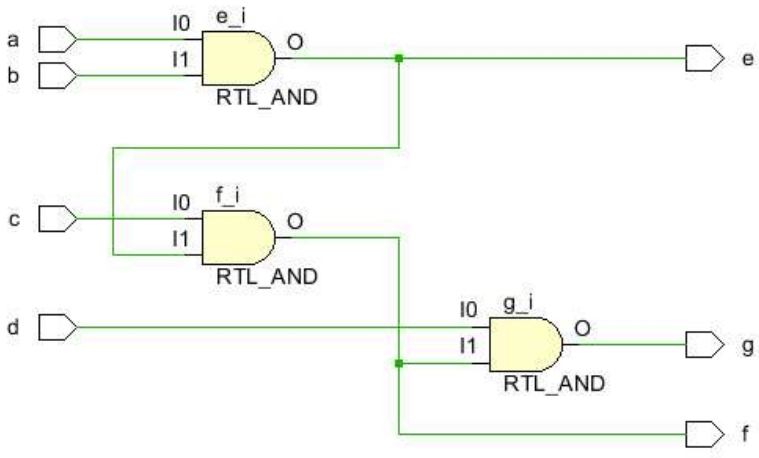
3_20191621_four_input_and_b.v	2_20191621_four_input_and_b_tb.v

`timescale 1ns / 1ps
module
3_20191621_four_input_and_gate_b(
input a,
input b,
input c,
input d,
output e,
output f,
output g
);
assign e = a&b;
assign f = c&e;
assign g = d&f;
endmodule
	`timescale 1ns / 1ps
	
	module
	3_20191621_four_input_and_gate_b_tb;
	reg aa;
	reg bb;
	reg cc;
	reg dd;
	wire ee;
	wire ff;
	wire gg;
	
	3_20191621_four_input_and_gate_b
	u_3_20191621_four_input_and_gate_b(
	.a (aa),
	.b (bb),
	.c (cc),
	.d (dd),
	.e (ee),
	.f (ff),
	.g (gg)
	);
	initial aa = 1'b0;
	initial bb = 1'b0;
	initial cc = 1'b0;
	initial dd = 1'b0;

	always aa = #200 ~aa;
	always bb = #100 ~bb;
	always cc = #50 ~cc;
	always dd = #25 ~dd;
	
	initial begin
	#1000
	$finish;
	end
	endmodule



<Simulation>

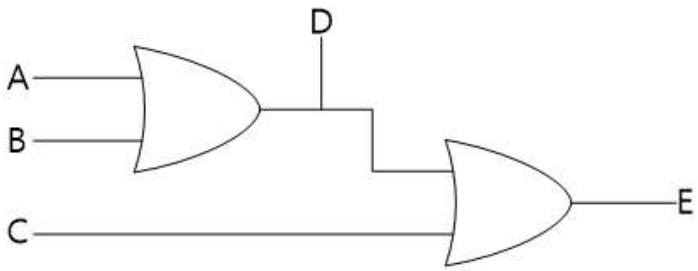


e는 a&b값이 출력되고 f는 c&e 값이 출력되고 g는 d&f 값이 출력된다. 결과적으로 g는 a,b,c,d값이 모두 1일때만 1로 출력된다는 것을 simulation을 통해서 알 수 있다.

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

5. 3-input OR gate의 simulation 결과 및 과정에 대해서 설명하시오.

(3 input, 2 output)[3장 ppt 35 page 참조, 진리표 작성]



<3 input - 2 output>

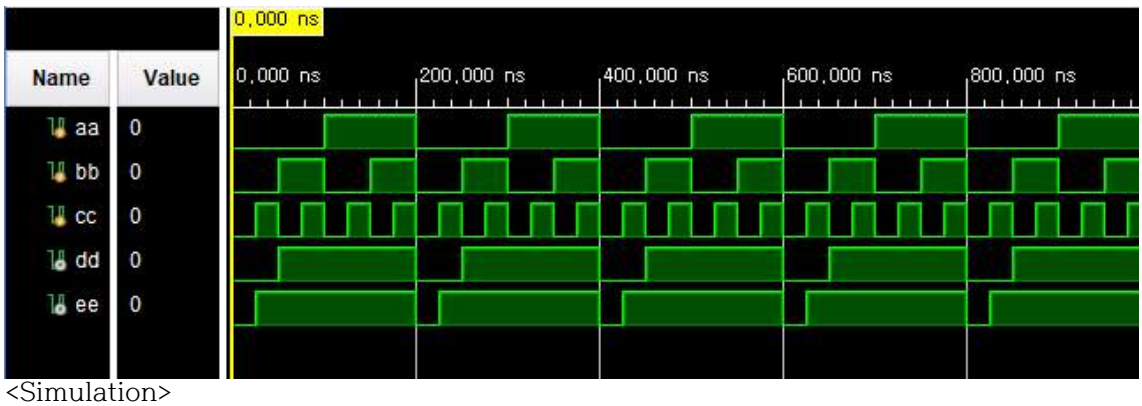
3_20191621_three_input_or_b.v	3_20191621_three_input_or_b_tb.v
`timescale 1ns / 1ps module 3_20191621_three_input_or_gate_b(input a, input b, input c, output d, output e);	`timescale 1ns / 1ps module 3_20191621_three_input_or_gate_b_tb; reg aa; reg bb; reg cc; wire dd; wire ee;

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

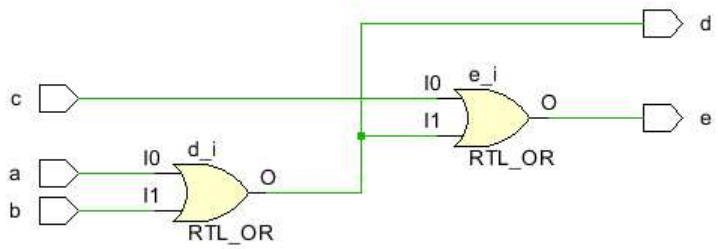
```
3_20191621_three_input_or_gate_b
u_3_20191621_three_input_or_gate_b(
.a (aa),
.b (bb),
.c (cc),
.d (dd),
.e (ee)
);
initial aa = 1'b0;
initial bb = 1'b0;
initial cc = 1'b0;

always aa = #100 ~aa;
always bb = #50 ~bb;
always cc = #25 ~cc;

initial begin
        #1000
        $finish;
```



<Simulation>

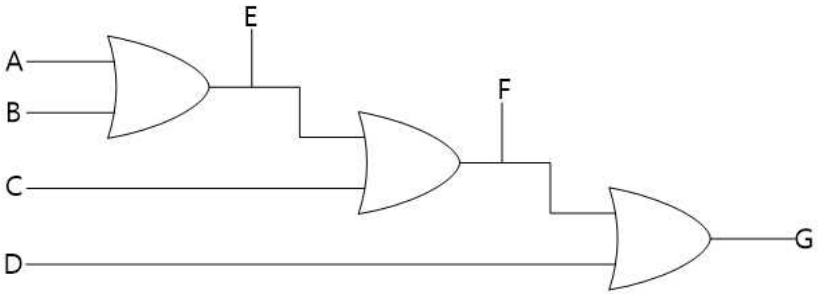


d는 a|b의 값이 출력되고 e는 c|d의 값이 출력된다. 결과적으로 e는 a,b,c가 모두 0인 경우를 제외하고는 항상 1이 출력되는 것을 simulation을 통해서 알 수 있다.

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

6. 4-input OR gate의 simulation 결과 및 과정에 대해서 설명하시오.

(4 input, 3 output)[3장 ppt 37 page 참조, 진리표 작성]

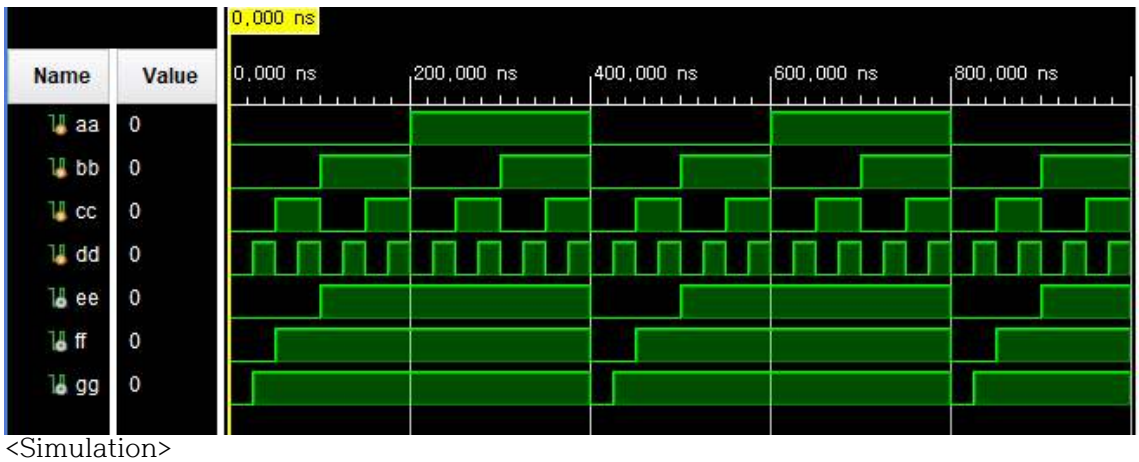


<4 input - 3 output>

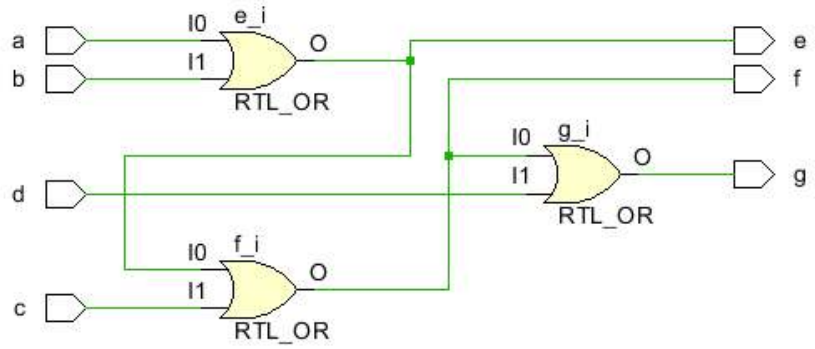
3_20191621_four_input_or_b.v	3_20191621_four_input_or_b_tb.v

`timescale 1ns / 1ps
module
3_20191621_four_input_or_gate_b(
input a,
input b,
input c,
input d,
output e,
output f,
output g
);
assign e = a|b;
assign f = e|c;
assign g = f|d;
	`timescale 1ns / 1ps
	
	module
	3_20191621_four_input_or_gate_b_tb;
	reg aa;
	reg bb;
	reg cc;
	reg dd;
	wire ee;
	wire ff;
	wire gg;
	
	3_20191621_four_input_or_gate_b
	u_3_20191621_four_input_or_gate_b(
	.a (aa),

endmodule	.b (bb),
	.c (cc),
	.d (dd),
	.e (ee),
	.f (ff),
	.g (gg)
	);
	initial aa = 1'b0;
	initial bb = 1'b0;
	initial cc = 1'b0;
	initial dd = 1'b0;
	
	always aa = #200 ~aa;
	always bb = #100 ~bb;
	always cc = #50 ~cc;
	always dd = #25 ~dd;
	
	initial begin
	#1000
	$finish;
	end
	endmodule



<Simulation>



e는 a|b 값이 출력되고 f는 e|c 값이 출력되고 g는 d|f값이 출력된다. 결과적으로 g의 경우에는 a,b,c,d가 모두 0인 경우에만 0이 출력되고 나머지 값은 모두 1이 출력되는 것을 simulation을 통해서 알 수 있다. 이외의 경우에도 진리표에 맞추어서 값이 나오는 것을 확인할 수 있다.

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

<진리표>

7. 결과 검토 및 논의사항

simulation을 통한 결과들이 진리표와 동일하게 나오는 것을 확인해볼 수 있었다. 각각의 경우를 판단하기 위해서 시간을 조절하여서 상황에 맞는 경우의 수를 조합할 수 있도록 하였다. 또한, output을 다르게 했을 때에도 최종적인 결과는 동일하게 나온다는 것을 알 수 있었다.

8. 추가 이론 조사 및 작성

논리회로를 설계할 때는 논리식과 진리표가 이용된다. 좀 더 회로도적인 표기를 하기 위해서 MIL기호 등의 논리 소자 기호가 이용되기도 한다. 1960년대 표준 논리 IC가 등장한 이후에는 아날로그 회로 설계와 논리 설계를 따로 분리해서 구현할 수 있게 되었다. 1990년대 후반 쯤에 되어서는 논리 회로 프로그램을 이용하여 FPGA를 이용하였다.

논리 회로를 접근할 수 있는 방식은 조합 논리 와 순차 논리가 있다. 조합 논리란 시간의 개념이 없는 논리 회로이며, 순차 논리란 시간의 개념까지 포함된 논리 회로 이다.

이번 실험에서는 AND, OR 각각으로만 진행하였으나, 여러 연산이 합쳐져 있을 경우에는 연산의 우선순위에 따라서 계산되게 된다. 일반적인 수식에서와 같이 논리 곱인 AND가 논리 합인 OR보다 연산의 순위가 높다. 또한 부정 연산인 NOT 연산이 AND와 OR보다 연산 우선순위가 높다. 즉 연산의 우선순위는 NOT>AND>OR 이다.