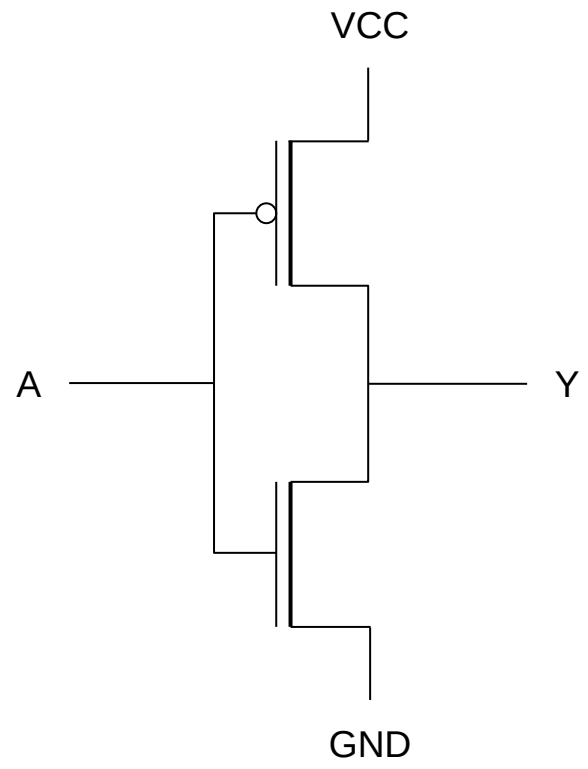
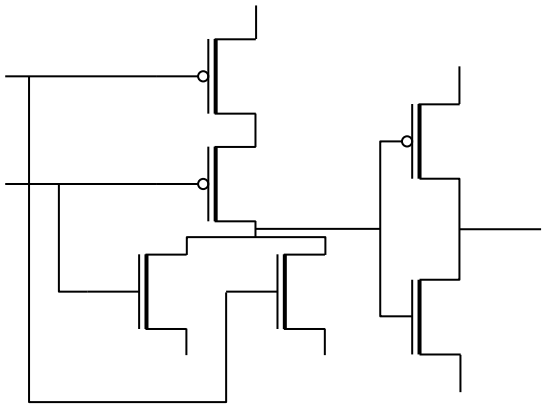
3주차 예비보고서

전공: 컴퓨터공학 학년: 2학년 학번: 20191629 이름: 이주헌

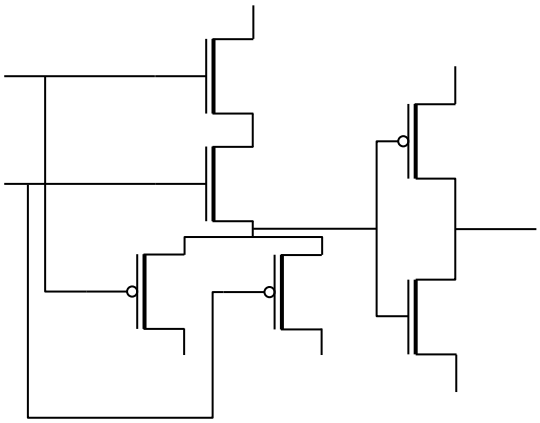
1. 논리게이트 AND/OR/NOT의 구조를 transistor-level로 그리시오.



NOT 게이트 (인버터)



OR 게이트 (NOR 게이트에 인버터를 추가한 모양)



AND 게이트 (NAND 게이트에 인버터를 추가한 모양)

2. AND/OR/NOT logic의 특성을 조사하시오.

AND 게이트와 OR 게이트는 모두 두 개의 입력을 받아 하나의 값을 출력하는 논리 게이트이다. AND 게이트는 두 입력 모두가 논리적 1일 때만 1을 출력하지만, OR 게이트는 두 입력 중 하나라도 논리적 1이면 1을 출력한다. 두 게이트의 진리표는 다음과 같다.

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

왼쪽: AND 게이트의 진리표, 오른쪽: OR 게이트의 진리표

NOT 게이트는 하나의 입력을 받아 하나의 값을 출력하는 논리 게이트인데, 단순히 받은 논리 입력을 반전하여 출력하는 게이트이다. 즉, 1을 입력 받으면 0을 출력하고, 0을 입력 받으면 1을 출력하게 된다. 보통 논리적 AND 게이트와 OR 게이트는 NAND 게이트와 OR 게이트에 NOT 게이트를 연결하여 구성된다.

A	Y

0	1
1	0

NOT 게이트의 진리표

3. Fan-out에 대하여 조사하시오.

Fan-out은 어떤 논리 소자의 출력이 다른 소자의 입력으로 들어갈 때의 개수 제한이다. 논리 회로를 구성할 때 여러 논리 소자를 연결해서 구성하게 되는데, 이 때 보통 하나의 논리 소자의 출력을 다른 여러 논리 소자의 입력과 연결하는 경우가 흔하다. 어떤 소자의 fan-out이라고 함은 그 소자의 출력에 연결할 수 있는 소자의 최대 개수이다. 만약 fan-out 이상의 소자를 연결한다면 논리회로의 안정성과 신뢰성을 보장할 수 없게 된다.

Fan-out이 존재하는 이유는 크게 두 가지이다. 먼저 현실적으로 안전하게 어떤 논리 소자를 거쳐 보낼 수 있는 전류량은 한계가 있다. 하나의 출력에 여러 개의 입력을 연결하게 되면 입력의 개수만큼 전력을 많이 사용하게 되는데, 너무 많은 입력을 연결하여 과부하를 일으키면 소자가 손상될 가능성이 있다. 또한, 소자가 손상되지 않더라도 많은 입력으로 인한 전압 강하는 입력 소자로 들어가는 논리 상태의 신뢰도를 떨어트린다.

또, 여러 개의 소자를 연결하면 회로의 길이가 늘어나고, 자연히 정전용량이 늘어나 전파 지연 시간이 늘어나게 된다. 전파 지연이 일정 시간 길어지게 되면 타이밍에 민감한 부품이 요구하는 시간적 제약 조건에 맞추지 못 할 가능성이 있다. 이로 인해 race condition이 발생하여 고치기 힘든 버그가 발생하거나 취약점으로 악용될 가능성이 있다.

4. 전파 지연에 대하여 조사하시오.

전파 지연은 논리 회로에 신뢰성 있는 입력이 주어진 뒤, 신뢰성 있는 출력이 도출될 때 까지의 시간이다. 논리 회로의 전파 지연은 회로를 구성하고 있는 각 소자의 전파 지연 시간을 전부 합한 것이다. 현실적으로 전파 지연을 없애는 것은 불가능하기 때문에, 전파 지연 시간을 잘 고려해서 회로를 작성해야 한다. 만약 전파 지연 시간이 과도하게 길어질 경우, 위에서 설명한 것과 같이 race condition이 발생하여 버그가 생기는 경우가 있다.

5. Verilog의 task 및 function에 대하여 조사하시오.

Verilog에서 task와 function 모두 여러 문(statement)를 모아 하나의 그룹으로 묶는 것이라는 점에서 비슷하지만, task는 순차적으로 코드를 실행시키거나 타이밍 지연을 포함할

수 있다. 또, task는 임의의 입력과 출력을 가질 수 있다. 따라서 task는 순차적 논리 회로와 결합 논리 회로를 모델링하는 데 모두 사용될 수 있다.

```
task read_memory;
  input [7:0] address;
  output [7:0] data;
  begin
    $display("Mounting values onto the memory bus...");
    addr = address;
    ce = 1;
    rd = 1;
    @(posedge clk);
    $display("Reading data from the bus...");
    data = data_rd;
    @(negedge clk);
    $display("Resetting data bus...");
    addr = 0;
    ce = 0;
    rd = 0;
    $display("Data: %h", data);
  end
endtask
```

Verilog의 function은 task와 달리 순차적으로 코드를 실행시킬 수 없으며, 시간 지연을 포함할 수 없다. 따라서, posedge, negedge와 같은 시간 지연을 포함하는 문의 작성이 불가능하다. 또한, function은 임의의 입력과 단 하나의 출력만을 가질 수 있다. 이러한 특성 때문에 function은 문(statement)뿐만이 아니라 식(expression)으로도 사용될 수 있어 값으로 사용할 수 있다는 장점이 있다. 즉, function은 결합 논리 회로를 모델링하는 데 사용될 수 있다.

```
function clamp_address;
  input [7:0] address;
  reg [7:0] offsetted;
  begin
    offsetted = address + 4'b1000;
    clamp_address = offsetted & 8'b01111111;
  end
endfunction
```