

## 2주차 결과보고서

전공: 경영학과

학년: 4학년

학번: 20190808

이름: 방지혁

1.

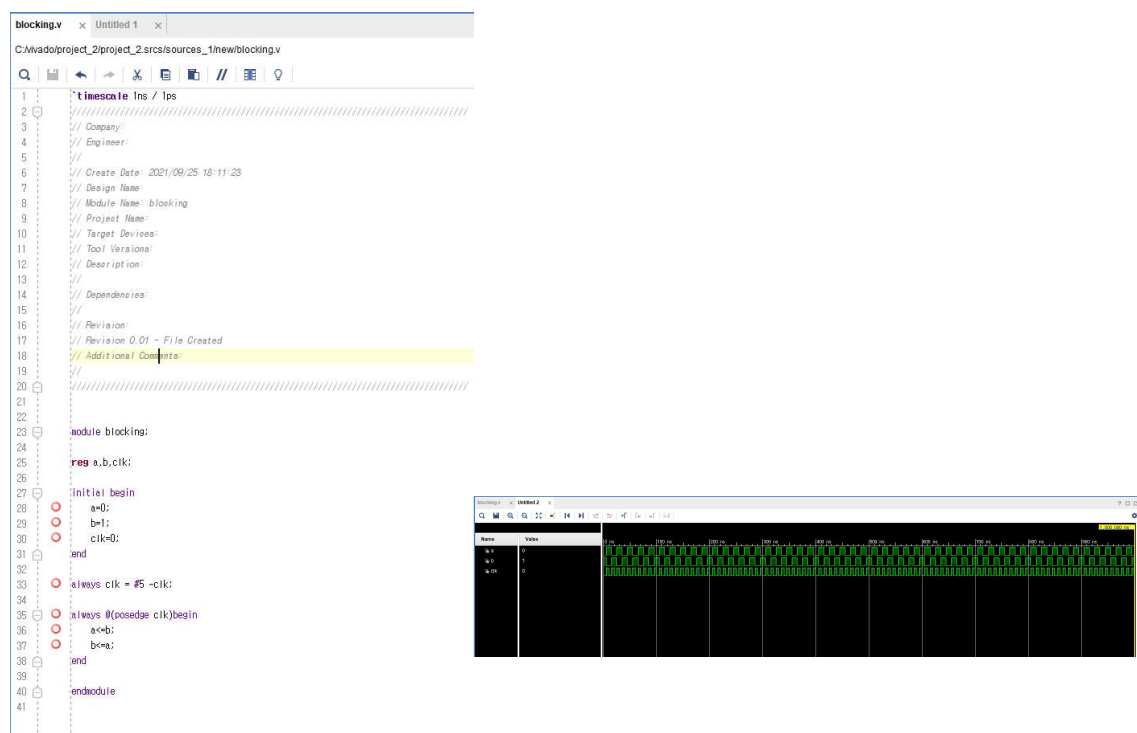
Verilog의 할당문 종류에는 연속 할당문과 절차형 할당문이 있습니다.

첫 번째로, 연속 할당문 (Continuous Assignment)은 net 자료형에 값을 할당할 때 사용하는 것입니다. combinational logic을 사용하며 우변에 위치해 있는 피연산자의 값이 변할 때마다 값이 net형 객체에 값을 할당합니다. assign 구문과 deassign 구문이 존재하는데, 둘이 서로 상반된 기능을 한다고 보면 됩니다. 또한, always 블록 안에서 사용할 수 없습니다.

반면 절차형 할당문 (Procedural Assignment)은 always 블록 안에서 사용됩니다. 이 절차형 할당문은 블로킹 할당과 논블로킹 할당으로 나뉩니다.

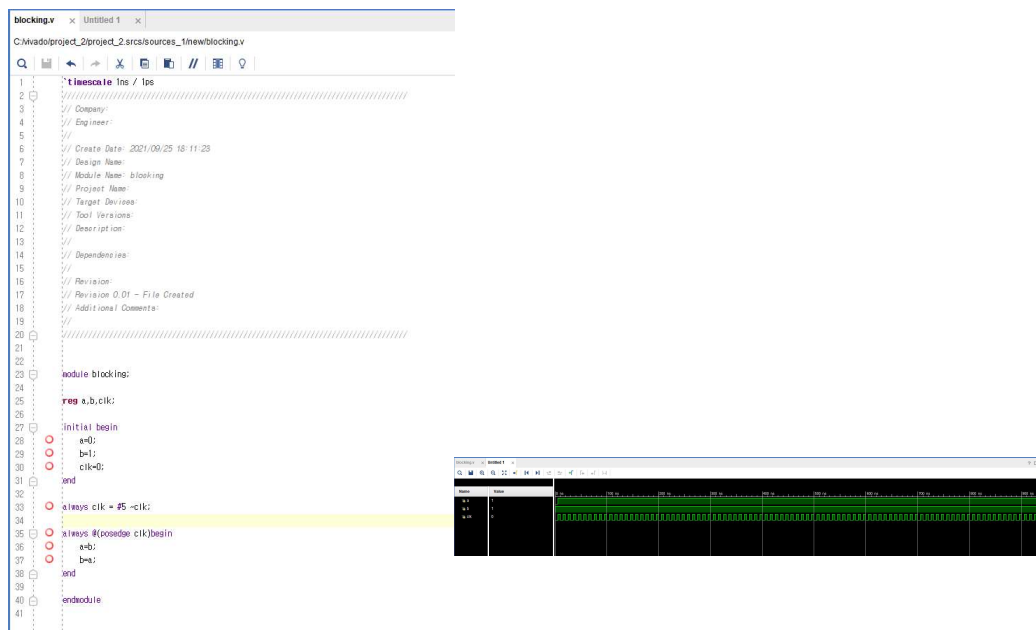
2.

blocking assignment는 할당 연산자 =을 사용합니다. 현재 줄의 statement가 수행되는 동안 block 처리가 되기 때문에 다음 statement를 수행하지 않습니다. 이후 업데이트된 값을 사용합니다. 반면 non-blocking assignment는 할당 연산자 <=를 사용하며 줄 순서와 무관하게 동시에 여러 statement를 수행합니다.



```
1 // timescale 1ns / 1ps
2 //////////////////////////////////////////////////
3 // Company:
4 // Engineer:
5 //
6 // Create Date: 2021/09/25 18:11:23
7 // Design Name:
8 // Module Name: blocking
9 // Project Name:
10 // Target Devices:
11 // Tool Versions:
12 // Description:
13 //
14 // Dependencies:
15 //
16 // Revision:
17 // Revision 0.01 - File Created
18 // Additional Comments:
19 //
20 //////////////////////////////////////////////////
21
22 module blocking;
23
24     reg a,b,clk;
25
26     initial begin
27         a=0;
28         b=1;
29         clk=0;
30     end
31
32     always clk = #5 ~clk;
33
34     always @(posedge clk)begin
35         a<=b;
36         b<=a;
37     end
38
39 endmodule
40
41
```

위의 캡처본은 non blocking assignment입니다. a와 b가 번갈아 가며 값을 반복하게 됩니다.



위의 캡처본은 blocking assignment입니다.

보시다시피 b 값에 할당하는 변화가 a 값이 할당한 후에 일어납니다. 그렇기에 계속 일정한 값을 유지하게 됩니다.

### 3.

우선 Verilog는 하드웨어 프로그래밍 언어, C 언어는 소프트웨어 프로그래밍 언어라는 측면에서 근본적으로 차이점이 존재하지만, for문, if문, while문, case문은 비슷하게 작동합니다. C언어의 경우 for문은 이렇게 작성해야 합니다.

```
for (int i = 0; i < 10; i++) {
    // 반복 수행할 코드
}
```

반면, verilog의 경우 {}로 반복할 코드를 감싸는 것이 아니라 begin과 end로 감싸야 합니다.

```
for (i = 0; i < 10; i = i + 1) begin
    // 반복할 코드
end
```

C언어의 경우 if문은 이렇게 작성해야 합니다.

```
if (a < b) {
    // a가 b보다 작으면 실행
}
```

반면, verilog의 경우 {}로 실행할 코드를 감싸는 것이 아니라 begin과 end로 감싸야 합니다.

```
if (a < b) begin
    // a가 b보다 작으면 실행
end
```

C언어의 경우 while문은 이렇게 작성해야 합니다.

```
while (i < 100) {
```

```

        // 반복 수행할 코드
        i++;
    }

```

반면, verilog의 경우 {}로 실행할 코드를 감싸는 것이 아니라 begin과 end로 감싸야 합니다. while 문이 존재하지만, 하드웨어적인 제약 때문에 조심해야 합니다. 그렇기에, while 문은 시뮬레이션에서만 사용 가능하고, 합성할 경우에는 잘 사용하지 않습니다.

```
while (i < 100) begin
```

```
    // 반복할 코드
```

```
    i = i + 1;
```

```
end
```

C언어의 경우 case 문은 이렇게 작성해야 합니다.

```
switch (input) {
```

```
    case 1:
```

```
        // 코드
```

```
        break;
```

```
    case 2:
```

```
        // 코드
```

```
        break;
```

```
    default:
```

```
        // 코드
```

```
        break;
```

```
}
```

Verilog에서는 이렇게 사용합니다.

```
case (input)
```

```
    1: begin
```

```
        // 코드
```

```
    end
```

```
    2: begin
```

```
        // 코드
```

```
    end
```

```
    default: begin
```

```
        // 코드
```

```
    end
```

```
endcase
```

멀티플렉서처럼 여러 조건을 비교하여 그에 맞는 동작을 선택할 때 사용됩니다. C와 달리 Verilog는 동시적으로 여러 조건을 처리하기 때문에 하드웨어적인 측면에서 보면 병렬 처리가 되는 것입니다. 또한, Verilog의 for문과 while문은 컴파일 시점에서만 동작합니다.

4.

Verilog에는 크게 두 가지 자료형, net과 variable이 존재합니다. 그중 net 자료형은 논리 게이트, 모듈 등의 하드웨어 요소들 사이의 물리적인 연결을 추상화한 것입니다. 연속 할당문, 게이트 프리미티브와 같은 구동자의 값에 의해 값이 연속적으로 유지됩니다. 단, trireg 외에는 신호 값을 저장하지는 않습니다. 구동자가 연결되지 않으면, default 값이 z, 즉 high impedance가 됩니다. default 초기값은 z로설정 되어 있습니다. net 값이 변하면 자동으로 새로운 값이 저장됩니다. net의 자료형에는 wire, tir, supply0, supply1, tri0, tri1, wand, wor, triand, trior, trireg가 존재합니다.