실험 Verilog-1: 결과보고서

전공: 컴퓨터공학 학년: 2 학번: 20171645 이름 박찬우

1. 목적

Verilog에 대해 이해한다.

2. 요구 사항

1) 연속 할당문, 절차형 할당문의 차이를 비교하여 설명하시오.

연속 할당문, 절차형 할당문 모두 객체에 값을 주는 방법이라는 공통점이 있지만, 차이점 역시 존재한다. 연속 할당문의 경우 assign 문을 이용해 net형 자료형에 값을 할당하는데, assign 구문은 입력 피연산자의 값에 변화가 발생하면 우변의 식이 평가되고, 결과가 주변의 net형 자료형을 구동하는 하드웨어적 특성을 갖게 된다. 쉽게 말하면, assign 문의 우변의 값에 변화가 생기면 좌변의 객체에 할당이 일어나는 방식이다.
반면 절차형 할당문은 variable 자료형 객체에 값을 할당하는데, always 구문이나 initial 구문 내에서 사용된다. reg 자료형과 메모리 변수에 값을 갱신하는데, 이때 다음 절차형 할당문 전까지 변수에 할당된 값을 유지한다는 차이가 있다. 즉, 할당문의 순서가 결과에 영향을 미칠 수 있다는 것이다. 절차형 할당문은 크게 Blocking statement와 NonBlocking statement로 나뉜다.

2) Blocking nonBlocking 문법의 차이를 simulation을 통해 설명하시오.

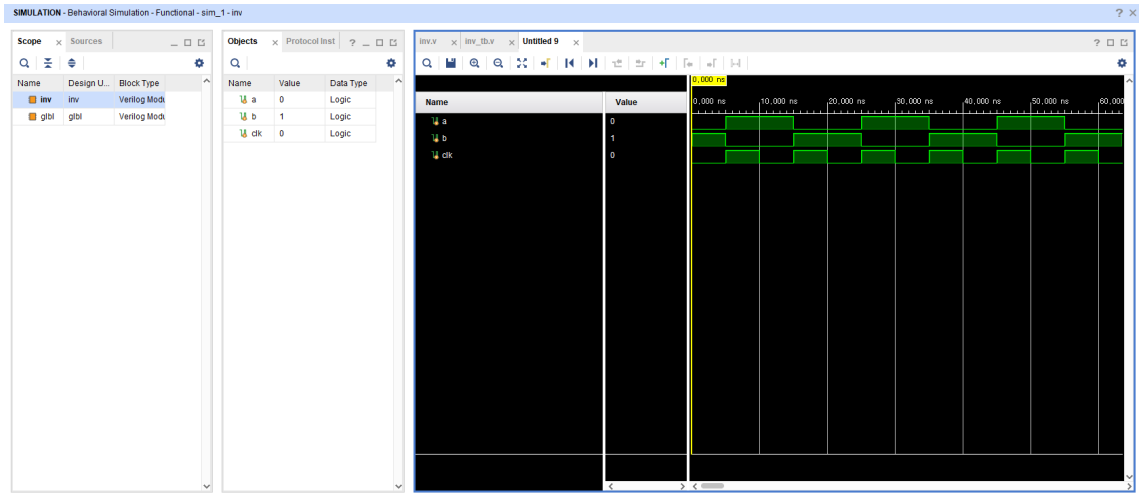
Blocking 문법은 현재 할당문의 실행이 끝난 뒤 다음 할당문이 실행된다. 반면, nonBlocking 문법은 나열된 순서와 상관없이 우변이 동시에 평가된 후 정해진 스케줄에 의해 값을 갱신한다.

차이점을 명확히 보여주는 예제는 다음과 같다.

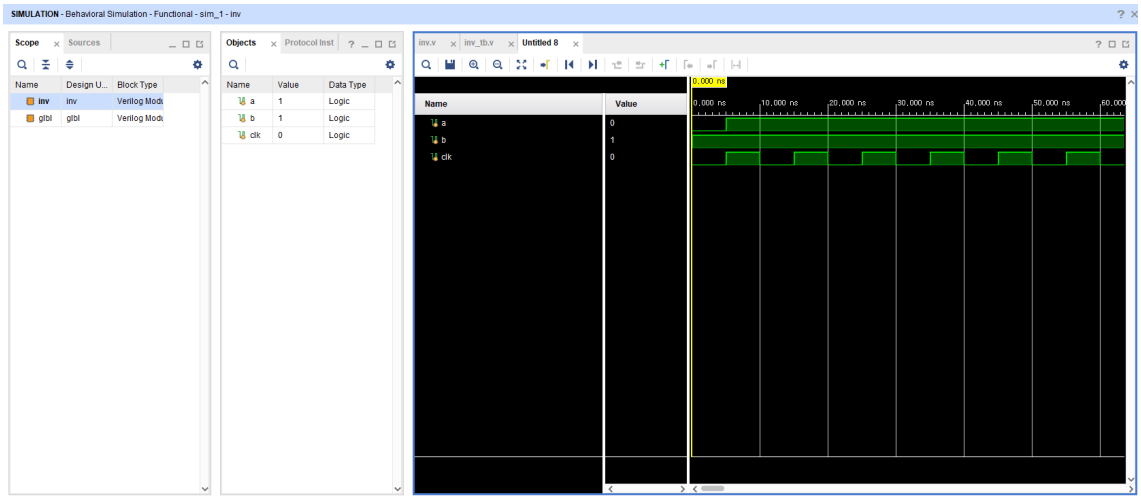
```
1  `timescale 1ns / 1ps
2
3  module inv;
4
5  reg a,b,clk;
6
7  initial begin
8      a=0;
9      b=1;
10     clk=0;
11 end
12
13 always clk = #5 ~clk;
14
15 always @(posedge clk)begin
16     a<=b;
17     b<=a;
18 end
19
20 endmodule
```

```
1  `timescale 1ns / 1ps
2
3  module inv;
4
5  reg a,b,clk;
6
7  initial begin
8      a=0;
9      b=1;
10     clk=0;
11 end
12
13 always clk = #5 ~clk;
14
15 always @(posedge clk)begin
16     a=b;
17     b=a;
18 end
19
20 endmodule
```

좌측은 <= 구문을 사용한 Nonblocking statement의 예시이고, 우측은 Blocking statement의 예제이다. 다음 두 코드를 시뮬레이션 해보면 다음과 같은 결과를 얻을 수 있다.



<nonblocking>



<blocking>

nonblocking statement에서는 a<=b 와 b<=a 구문이 동시에 실행되어 값이 서로 바뀌는 반면, blocking statement에서는 a=b 구문이 먼저 실행된 후 b=a 구문이 실행되어 값이 고정되어 있는 걸 확인할 수 있다.

3) Verilog 의 for문, if문, while문, case문 을 C언어와 비교하여 설명하시오.

Verilog의 for, if, while, case 등은 모두 C의 for, if, while, switch문과 유사하게 사용한다. 가장 큰 차이로 C에서는 대괄호('{' , '}')를 사용하는 반면 Verilog에서는 begin 과 end를 사용한다는 점이다. 그 외에는 거의 차이점이 없는데, 이들 구문을 사용하는 각각에 대한 예제는 다음과 같다.

<if의 경우>

C의 경우

```
 if (enable == '1') {
        data = 10;
}
```

Verilog의 경우

```
if (enable ==1'b1 ) begin
        data = 10;
end
```

<case의 경우>

C의 경우

```
switch(data) {
```

```
        case 0 :
                printf("data is 0"); break;
        case 1 :
                printf("data is 1"); break;
        default :
                printf("data is not binary");
}
```

Verilog의 경우

```
case(data)
        0 : $display("data is 0");
        1 : $display("data is 1");
        default : $display("data is not binary");
endcase
```

<while의 경우>

C의 경우

```
while(flag) {
~
}
```

Verilog의 경우

```
while(flag) begin
~
end
```

<for의 경우>

C의 경우

```
for(i=0; i<16; i=i+1)
{
        ~
}
```

Verilog의 경우

```
for(i=0; i<16; i=i+1)
begin
        ~
end
```

4) Verilog의 net 형 자료형에 대해서 조사하시오.

verilog에서 net형 자료형은 logic gate나 module 등 하드웨어 요소들 사이에서의 물리적 연결을 나타내기 위해 사용하는 자료형으로, 대표적으로 단순 연결을 위해 쓰는 wire 등이 있다. net형 자

료형은 연결형 데이터 타입으로, 만약 구동자가 연결되지 않았으면 기본값인 z가 된다. 이때 z는 driver가 없는 high-impedance 상태를 나타내는데, 간단히 말하면 전기적으로 절연된 상태를 나타낸다.