2주차 결과보고서

전공 : 국제한국학과 학년 : 4학년 학번 : 20181202 이름 : 김수미

**1. 연속 할당문, 절차형 할당문의 차이를 비교하여 설명하시오.**

할당문이란 객체에 값을 할당하는 방법으로, 일반적인 프로그래밍 언어에서는 할당문 좌변의 변수에 우변의 값을 할당할 수 있다. Verilog 언어에서는 연속 할당문과 절차형 할당문 총 두가지 유형의 할당문이 존재한다.

**1) 연속 할당문**

연속 할당문은 net 자료형 변수에 특정 논리값을 할당할 때 사용된다. net 자료형 변수란 디바이스의 물리적인 연결을 나타내는 변수로 wire, tri 등이 여기에 해당된다.

**assign** : assign 구문은 입력 피연산자의 값에 변화가 발생할 때 마다 우변의 식이 평가되고, 그 결과갑이 할당문 주변의 net 변수를 구동시키는 하드웨어적 특성을 갖는다.

**deassign** : 변수에 대한 assign문의 영향을 제거할 때 사용된다.

연속 할당문을 사용하면 게이트 기본요소만을 사용할 때 보다 더 추상적이고 복잡한 조합 논리 회로를 설계할 수 있으며, 코드의 길이도 줄일 수 있다.  
연속 할당문은 절차형 할당문과 달리 procedure 내부에서 사용할 수 없다.

**2) 절차형 할당문**

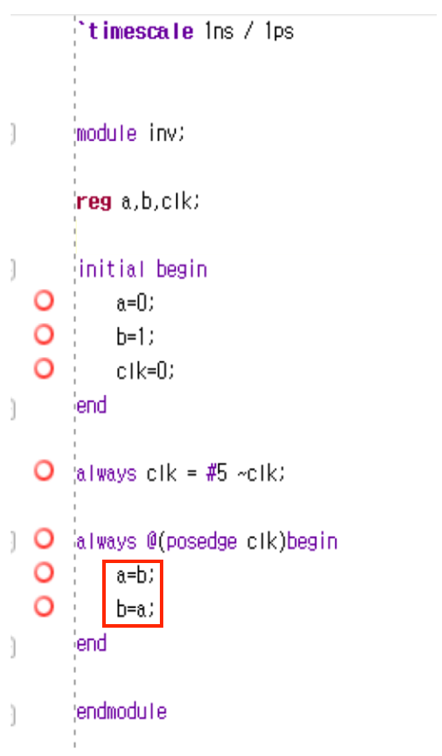
절차형 할당문은 reg 변수 또는 메모리 변수 등 variable형 객체에 값을 할당할 때 사용된다. 연속 할당문과 달리 always 나 initial 구문 같은 procedure 내부에서 사용할 수 있으며 절차형 할당문에 의해 한번 값이 할당되면 다음 절차형 할당문에 의해 값이 갱신될 때 까지 변수에 할당된 값이 유지된다. 이런 방식을 이용하면 변수에 값을 저장하여 사용할 수 있다. Blocking statement 와 Non-blocking statement가 절차형 할당문에 해당된다.

**2. Blocking 및 Non-Blocking문법의 차이를 simulation을 통해 설명하시오.**

**1) Blocking statement**

Blocking statement는 절차형 할당문의 한 종류로, 할당 기호로 = 을 사용한다. Begin 부터 end 까지 Line by Line의 순서로 계산과 동시에 저장이 이루어진다. 즉 변수에 값을 할당까지 완료한 후 다음 line의 할당문으로 넘어간다. 한 문장의 수행이 끝나기 전에 blocking 되기 때문에 blocking statement라고 불린다.

**#t (변수) = (연산)** : t의 시간 후 연산하고 연산 결과 값을 변수에 할당한다는 의미



왼쪽은 Blocking statement의 사용 예시이다.

a = b, b=a 식으로 값을 할당했기 때문에 being ~ end 구문 안에서 변수 값이 갱신된다.

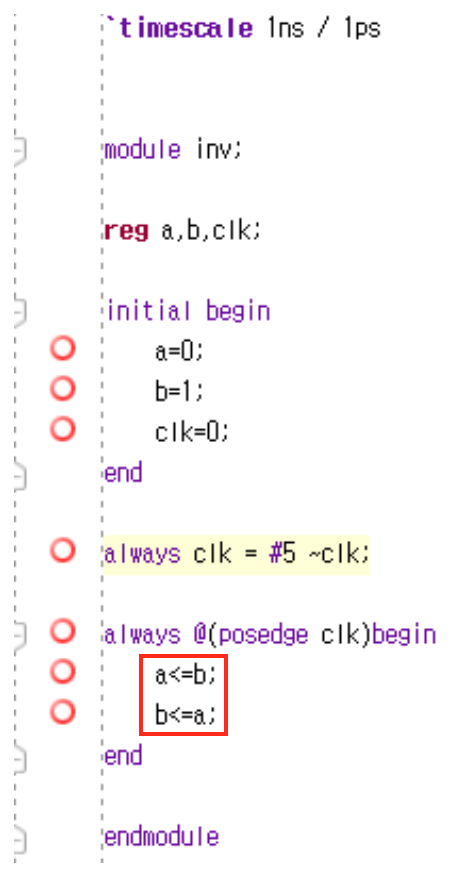
즉 bigin ~ end 구문이 끝나면

첫번째 줄의 a = b 에 의해 a 의 값은 1이 되고,  
두번째 줄의 b = a에 의해 b의 값은 1이 된다.

**2) Non-Blocking statement**

Non-Blocking statement역시 절차형 할당문의 한 종류로, 할당 기호로 <= 을 사용한다. Begin 부터 end 까지의 모든 계산을 수행 한 다음에 한꺼번에 저장 작업이 수행된다. 즉 계산을 수행할 때 Begin ~ end 내부 변수들의 값은 end에 도달할 때 까지 바뀌지 않는다. Non-Blocking 구분이 여러개인 경우, 모두 동시에 평가된 후 할당된다. 주로 공통적인 사건 발생 후 여러 data를 동시에 전송하기 위해 사용된다.

**(변수) <= #t** **(연산)** : 연산 수행 후 t의 시간이 지난 다음에 변수에 연산 결과 값을 할당(변수 할당을 예약해 놓는 것)

왼쪽은 Non-Blocking statement의 사용 예시이다.

a <= b, b <= a 식으로 값을 할당했기 때문에 being ~ end 구문이 끝난 뒤에 변수 값이 갱신된다.

즉 bigin ~ end 구문이 끝나면

첫번째 줄의 a <= b에 의해 a 의 값은 1이 되고,  
두번째 줄의 b <= a에 의해 b의 값은 0이 된다.

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

**1) for**

|  |  |
| --- | --- |
| C 언어 | Verilog |
| for (i = 0 ; i < 50 ; i++) { A[i] = 1;  } | for (i = 0 ; i < 50 ; i = i + 1) begin A[i] <= 1;  end |

Verilog에서의 for 문도 C 언어와 마찬가지로 for(초기화 ; 종료 조건 ; 제어변수)의 구조를 사용한다. 단, C 언어와 달리 ++ 또는 -- 연산자는 사용할 수 없다. 또한 for 구문이 적용되는 범위를 표현할 때 중괄호 대신 begin-end 구문을 사용한다.

**2) if**

|  |  |
| --- | --- |
| C 언어 | Verilog |
| if (x == 0) { y = y - 1 }  else if (x == 1) { y = y + 1; }  else { y = x;  } | if (x == 1'b0) begin y <= y - 1'b1; end  else if (x == 1'b1) begin y <= y + 1'b1; end  else begin y <= x; end |

if 문도 C 언어와 동일하게 if (조건)의 형태로 사용하면 된다. if, else, else if, nested if 등을 사용할 수 있으며 수식이 한 줄만 있는 경우 begin-end는 생략해도 된다.

**3) while**

|  |  |
| --- | --- |
| C 언어 | Verilog |
| while (x==1){ x = x – 1; } | while (x == 1'b0) begin x = x – 1; end |

while 문도 C 언어와 동일하게 while (조건)의 형태로 사용하면 되고, 조건이 거짓이 되면 루프를 종료한다.

**4) case**

|  |  |
| --- | --- |
| C 언어 | Verilog |
| switch (x) {  case 0:  y = 10; break;  case 1:  y = 20; break;  case 2:  y = 30; break;  case 3:  y = 40; break;  default:  break;  } | case (x)  0 : y = 10;  1 : y = 20;  2 : y = 30;  3 : y = 40; endcase |

Verilog는 C 언어와 달리 case문에서 switch와 break를 사용하지 않는다. case (값)의 형태로 값에 따라 (값 : 명령문)을 실행하며, break 없이도 해당 case만 실행 후 종료된다. case문의 끝에는 endcase를 써주면 된다.

**4. Verilog 의 net 형 자료형에 대해서 조사하시오.**

1번 에서 설명한 것처럼 net 자료형 변수란 디바이스의 물리적인 연결을 나타내는 변수로 논리 게이트나 모듈 등의 하드웨어 요소들 사이의 물리적 연결을 나타내기 위해 사용된다. 연속 할당문, 게이트 프리미티브 등과 같은 구동자에 의해 net의 값이 연속적으로 유지되며, 변수에 값을 저장하지는 않는다. 구동자가 연결되지 않으면, 변수 값은 default 값인 high-impedance(z)가 되며, 이는 1비트의 wire 값을 가진다.  
net 자료형의 종류에는 wire, wire, tri, wand, wor, triand, trior, supply0, supply1, tri0, tri1, trireg 등이 있다.