2주차 결과보고서

전공: 기계공학과 학년: 3학년 학번: 20191820 이름: 김형준

**1.**

연속 할당문은 assign문을 사용하여 net형 변수에 값을 할당하는 문장을 의미한다.

우변 수식의 값에 변화(event)가 발생할 때 마다 좌변의 객체에 값이 할당된다.

따라서 연속 할당문의 결과값은 할당문 문장의 순서와 관계 없고, 할당문 주변의 net을 구동하는 하드웨어적 특성을 갖는다. 또한, 연속 할당문은 always, initial, task, function 등의 procedure 내부에는 사용할 수 없다. 연속 할당문은 논리식으로 표현된 조합회로의 모델링에 사용될 수 있다.

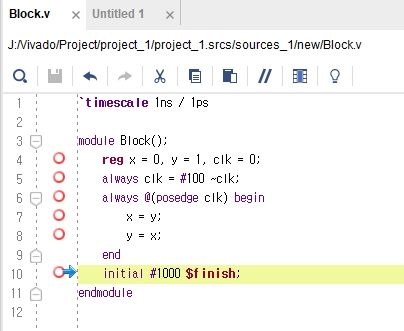
반면에, 절차형 할당문은 always, initial, task, function 등의 procedure 내부에서 우변 수식의 값을 좌변의 변수(reg, integer, time, real 등)에 값을 할당하는 문장을 의미한다.

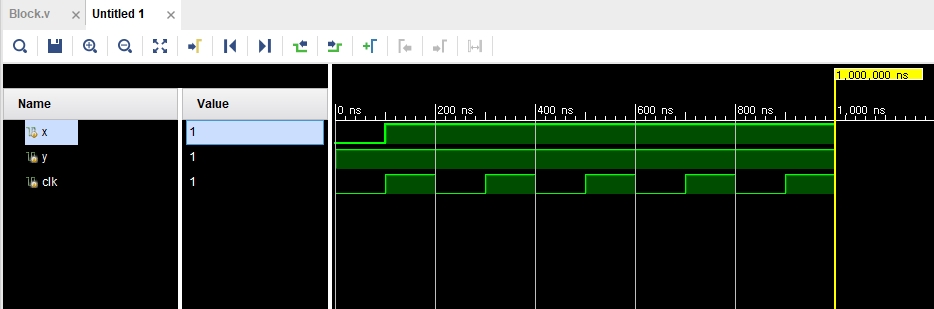
절차형 할당문은 문장이 나열된 순서대로 실행되며, 문장이 나열된 순서를 변경하면 결과값도 변할 수 있다. 따라서 절차형 할당문은 연속 할당문과 다르게 우변의 값의 변화와는 관계없이 문장의 실행에 의해 좌변에 값이 할당되는 소프트웨어적 특성을 갖는다. 절차형 할당문은 연산자로 =를 사용하는 Blocking statement와 연산자로 <=를 사용하는 Non blocking statement로 구분된다.

**2.**

Blocking statement는 연산자 기호로 =를 사용하며, procedure의 내부에서 순차적으로 한 줄씩 계산과 동시에 결과값의 할당을 수행한다. 수행과정에 지연시간이 없다.

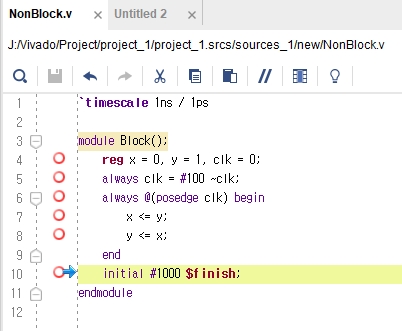
반면에, Non Blocking statement는 연산자 기호로 <=를 사용하며, procedure의 begin ~ end까지의 모든 계산을 수행 한 후 한꺼번에 결과값의 할당을 수행한다.

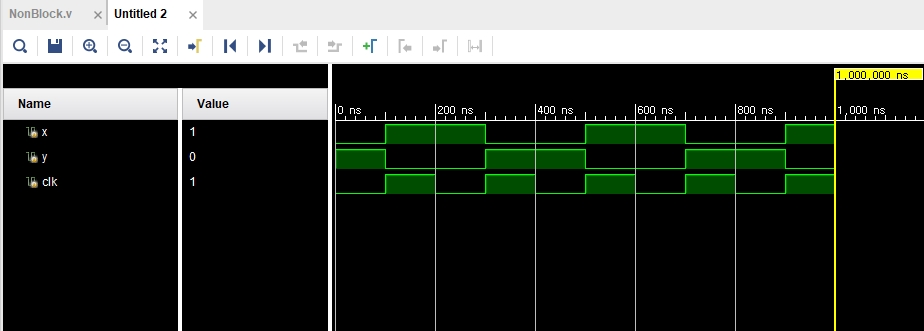




위는 Blocking statement의 예시 코드이다.

변수 x, y는 각각 초기값 0과 1을 가지고 있는데, 100ns부터 200ns 마다 x = y, y = x가 수행된다. 그러나, blocking statement로 할당문을 사용했기 때문에 우변의 결과값을 좌변에 할당하고 다음 줄로 넘어간다. 따라서 100ns 시점에 x = y에서 x에 1이 할당되고, y = x에서 y에 1이 할당되기 때문에 100ns 이후로는 x, y의 값이 1에서 변하지 않게 된다.





위는 Non Blocking statement의 예시 코드이다.

변수 x, y는 각각 초기값 0과 1을 가지고 있는데, 100ns부터 200ns 마다 x = y, y = x가 수행된다. 아까의 경우와 다르게, Non blocking statement로 할당문을 사용했기 때문에 우변의 결과값을 해당 블록 전체의 계산이 종료 된 후, 한번에 결과값을 할당한다.

따라서 100ns 시점에 x = y으로 x에 1이 할당되고, y = x으로 y에 0이 할당되고, (x, y값이 각각 1과 0이 됨) 그 다음 300ns 시점에 x = y으로 x에 0이 할당되고, y = x으로 y에 1이 할당된다. (x, y값이 각각 0과 1이 됨) 결과적으로 x, y의 값을 100ns부터 200ns마다 서로 바꾸는 형태가 된다.

이와 같이, 절자적 할당문에서 Blocking statement와 Non Blocking statement 중 어느 것을 사용하느냐에 따라 결과값이 변할 수 있다.

**3.**

1. for문

verilog의 for문은 C언어와 거의 동일하다.

for (초기화식 ; 조건식 ; 증감식) begin

반복할 문장

end

위와 같은 구조로 사용할 수 있다. C언어와 다른 점은 for문의 경계를 중괄호 대신 begin ~ end로 나타낸다는 점이다. 반복할 문장이 하나인 경우 begin과 end가 생략 가능하다.

2. if문

verilog의 if문은 C언어와 거의 동일하다.

if (조건식) begin

조건 만족시 실행할 문장

end

else begin

조건 불만족시 실행할 문장

end

위와 같은 구조로 사용할 수 있다. C언어와 다른 점은 if문의 경계를 중괄호 대신 begin ~ end로 나타낸다는 점이다. 실행할 문장이 하나인 경우 각각의 begin과 end가 생략 가능하다.

3. while문

verilog의 while문은 C언어와 거의 동일하다.

while (조건식) begin

반복할 문장

end

위와 같은 구조로 사용할 수 있다. C언어와 다른 점은 while문의 경계를 중괄호 대신 begin ~ end로 나타낸다는 점이다. 반복할 문장이 하나인 경우 begin과 end가 생략 가능하다.

4. case문

verilog의 case문은 C언어의 switch문과 유사하나 다소 차이를 보인다.

case (판정식)

항1 : 처리문1;

항2 : 처리문2;

…

항N : 처리문N;

default : 처리문 N+1;

endcase

위와 같은 구조로 사용할 수 있으며, 각 처리문이 한 줄을 넘어가는 경우, 해당 처리문을 begin ~ end로 묶어줘야 한다. c언어의 switch문과 유사하게 판정식의 결과값이 항 1~N중 일치하는 것이 있다면 해당 항에 연결된 처리문을 실행한다. 항 1~N중 일치하는 것이 없다면 default에 연결된 처리문을 실행한다. C언어와 다르게, 각 판정식의 마지막에 C언어의 break 같은 명령어를 따로 넣지 않아도 해당 처리문을 모두 실행하면 case문을 빠져나온다. 다른 구문들과 다르게 endcase를 사용해 case문을 끝낸다.

**4.**

Verilog의 net 자료형은 소자들간의 물리적인 연결을 추상화한 것으로써, 하드웨어 요소들 사이의 물리적인 연결을 나타내기 위해 사용한다. net형 자료형에 값을 할당할 때는 연속 할당문을 사용하며, default 자료형은 1비트의 wire이다.

ㆍ wire : 함축된 논리적 동작이나 기능을 갖지않는 단순한 연결을 위한 net

ㆍ tri : wire와 같이 단순한 연결을 위한 net이며, 하드웨어 3상태 (tri-state)에 사용됨

ㆍ wand : 다중 구동자를 갖는 net이며, and 연산을 하는 wire이다.

ㆍ wor : 다중 구동자를 갖는 net이며, or 연산을 하는 wire이다.

ㆍ triand : wand와 같으나, 하드웨어 3상태를 갖는다.

ㆍ trior : wor과 같으나, 하드웨어 3상태를 갖는다.

ㆍ supply0 : 회로접지에 연결되는 net

ㆍ supply1 : 전원에 연결되는 net

ㆍ tri0 : 저항성 pulldown에 의해 접지로 연결되는 net

ㆍ tri1 : 저항성 pullup에 의해 전원으로 연결되는 net

ㆍ trireg : 물리적인 net에 저장되는 전하를 모델링하는 net