2주차 결과보고서

전공: 컴퓨터공학과 학년: 2학년 학번: 20191619 이름: 이동석

**1. 연속 / 절차 할당문**

Assignments는 가장 nets과 variable에 값을 주는 기본적인 방법이다. 두가지 방법이 있는데, 첫 번째는 continuous(연속) 두 번째는 procedural(절차) 이다. 값의 할당은 기본적으로 = 로 이루어진다. 다음 표와 같은 값들이 등호의 기준으로 왼쪽에 올 수 있는 변수들이다. 오른쪽은 값으로 평가되는 모든 표현식이 올 수 있다. (스칼라 / 벡터)

테이블이(가) 표시된 사진

자동 생성된 설명

표를 보면 알 수 있듯, 연속 할당문은 nets 자료형에 값을 주는 것이다. 참고자료의 표현을 빌리면 gate가 net을 동작하는 것과 유사한 방식으로 net을 동작시킨다고 한다. 네트에 대한 할당은 연속적이고 자동적이다. 다음은 연속 할당문의 예시이다.

wire mynet ;

assign (strong1, pull0) mynet = enable ;

이 외에도 delay 구문이 조금 다르며 wire #10 wireA 와 같은 문법을 net delay라고 부른다.

절차형 할당문은 연속할당문과 다르게 variable에 값을 준다. 계속해서 값이 바뀌는 연속 할당문과 다르게, 지속의 개념이 없으며 다음 절차형 할당문이 오기전까지는 값을 유지한다. 절차형 할당은 always, initial, task 그리고 function으로 이루어진다. 주로, if구문이나 case 등을 사용해 제어한다. 다음은 절차형 할당문의 예이다.

reg[3:0] a = 4’h4 ;

같은 방법으로 initial을 사용하면, 다음과 같이 로 표현 가능하다. 그러나, 이때 상수만 와야하며 배열은 올 수 없다. (reg, integer, time, real, realtime의 자료형을 사용)

reg[3:0] a ;

initial a = 4’h4 ;

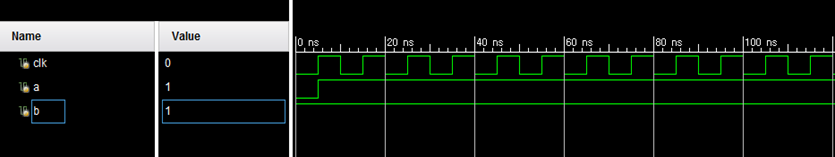
Procedural continuous assignments라는 두개를 혼합한 구문도 존재한다. assign 및 force 키워드를 사용하며, 표현식을 variable이나 net에 연속적으로 사용할 수 있다고 한다.

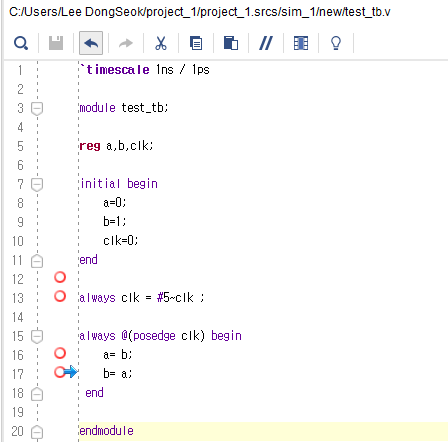
**2. Blocking과 NonBlocking의 차이 ( Simulation)**

Blocking procedural assignment의 예시는 다음과 같다. 일반적인 언어에서 사용하는 것과 같이 순차적으로 할당문이 실행되며 이전 할당문이 완료된 후 다음 할당문이 실행된다.

텍스트이(가) 표시된 사진

자동 생성된 설명

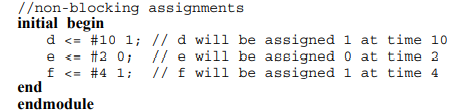
본래 직접 코딩을 통해 시뮬레이션을 구현하려 했으나, 아직 미숙하여 블로그와 강의자료를 참고해 결과를 가져왔다. 다음은 Blocking Statement의 코드와 그 결과이다.

코드 자체는 굉장히 단순하다.

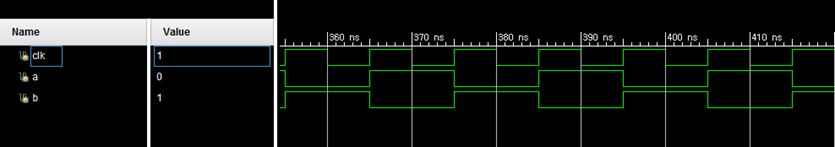
우선, initial은 한번만 실행되는 구문으로, a,b,clk에 각각 0,1,0을 할당한다.

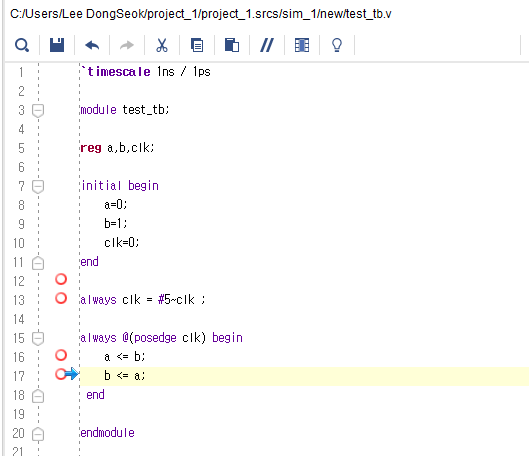
이후, always에 의해서 clk는 항상 5ns마다 ~(not)으로 인해 1과 0을 반복한다. 또한, Blocking이므로 a에 우선 b의 값인 1이 할당 되고 b 역시 a가 1이므로 1이 할당된다.

Non-blocking의 경우 예시는 다음과 같다. 앞선 blocking statement는 순서가 굉장히 중요하다. 하지만, non-blocking의 경우 같은 time에 명령문이 실행되므로 순서가 중요하지 않다. 이때, 표현식에서는 관계연산자 <=로 해석하지만 non-blocking구문에서는 할당 연산자로 해석된다.



마찬가지로 아래 코드를 통해 시뮬레이션 한 결과는 다음과 같다.



달라진 점은 =로 더 이상 할당하지 않고, <=를 이용해 할당한다.

이제 a와 b의 할당은 동시에 실행되기 때문에 기존 a=0 -> a=1 로 바뀌며 b=1 -> b= 0으로 바뀐다.

clk의 값은 앞서 설명한 것과 같다.

\*blocking이나 non-blocking 모두 절차형 할당문이다.

**3. Verilog와 C언어의 구문 차이**

C언어와 마찬가지로 Verilog에는 잠깐 설명했던 for이나 if-else, case같은 구문이존재한다. 가장 크게 다른점은, C언어에서는 { } 를 사용해 명령문의 관계를 명시하지만, **Verilog는 begin과 end**를 사용한다. 그러나 한줄만 올 경우 생략해도 무관한다. (각 명령문이 C에서 동작하는 방식은 생략하도록 하겠다.) 괄호의 차이 외엔 차이점은 없다고 봐도 무방하다.

우선 if-else의 경우 사용법은 다음과 같다.

텍스트이(가) 표시된 사진

자동 생성된 설명

다음은 case 이다. 이는 다중 결정 문으로 부른다. C와 마찬가지로 어떤 케이스도 만족하지 못하면 default가 실행된다. 그러나 C와는 다르게 break가 따로 존재하지 않으며, 자동으로 나오게 된다. 그러나, if-else-if문이 조금 더 일반적으로 사용된다. 또한, case문은 표현식에 x나 z가 있더라도 최종결과를 제공한다.

텍스트, 테이블이(가) 표시된 사진

자동 생성된 설명

다음은 loop이다. 반복문엔 대표적으로 for과 while이 있으며 Verilog 역시 당연히 존재한다. (이 외에도 forever과 repeat이 있다.) 기본적인 사용방법은 C언어와 동일하다. 다만, for에서는 C와 같은 i++나 ++i를 사용 할 수 없다. ( i=i+1 과 같이 사용)

텍스트이(가) 표시된 사진

자동 생성된 설명

**4. net자료형**

앞서 net 자료형에 간략하게 다룬적이 있다. Net이라는 자료형은 게이트와 같은 구조적 엔티티 간 물리적 연결을 나타낸다. Net의 default 초기화 값은 z 라고 한다. net-types는 다음 표와 같이 있다. trieg net을 제외하고는 값을 저장하지 않는다. 또한, trieg는 default로 x를 가진다.

테이블이(가) 표시된 사진

자동 생성된 설명

wire와 tri net은 구성요소(element)를 연결한다. 또한, 다중 구동자를 갖는 wand, wor, triand, trior 이 있으며 회로접지와 관련된 supply0, 전원과 연결되는 supply1, 저항성 pulldown\up에 의해 각각 접지나 전원으로 연결되는 tri0/1 이 있다. 마지막으로 trireg는 물리적인 net에 저장되는 전하를 모델링한다. (하드웨어 레지스터를 모델링) reg는 또한, 하드웨어 저장요소를 나타낼 필요가 없다.

**5. 참고문헌**

2-2강 - Verilog 기초 2 (기초 문법, 절차형 할당문) . (2020. 9. 8. 13:47). <https://intelligentcm.tistory.com/205>.

한국기술교육대학교 전기전자통신공학부 . (n.d.). <https://cms3.koreatech.ac.kr/sites/yjjang/down/dsys11/M01_VerilogHDL01.pdf>.

"IEEE Standard for Verilog Hardware Description Language," in IEEE Std 1364-2005 (Revision of IEEE Std 1364-2001) , vol., no., pp.1-590, 7 April 2006, doi: 10.1109/IEEESTD.2006.99495.