2주차 결과보고서

전공: 컴퓨터공학과 학년: 3학년 학번: 20191599 이름: 송경호

**1.**

verilog에서 할당문이랑, 객체에 값을 할당해주는 구문을 말한다. verilog에는 두가지 방식의 할당, 연속 할당문과 절차형 할당문이 존재한다.

**1-1) 연속할당문**

연속할당문은 wire, tri와 같이 디바이스의 물리적인 연결을 나타내는 net 자료형 변수에 논리값을 할당 시 사용된다. 연속할당문은 우변의 값에 변화가 있을 때마다 자동으로 좌변에 변화된 값에 대한 할당이 이루어진다. 다시 말해 연속할당문은 항상 모든 입력에 대한 출력을 계산하며 논리 레벨이 변할 때 이는 즉시 반영된다. assign구문과 deassign구문이 존재한다. assign 구문은 할당문 주변의 net 변수를 구동 시키는 하드웨어적 특정을 가지며 deassign 구문은 assign 구문의 영향을 제거할 때 사용된다.

연속할당문을 통해서 기본 요소만으로 구성한 게이트에 비해 복잡하고 수준 높은 논리 회로를 설계할 수 있으며 동시에 코드의 길이도 줄일 수 있다는 장점이 있다.

**1-2) 절차형 할당문**

절차형 할당문은 reg 변수에 값을 할당할 때 사용된다. 때문에 연속할당문과 달리 우변 수식 변화와는 무관하며 always와 initial 구문 같은 procedure 내부에서 사용 가능하다. Blocking 구문이 절차형 할당문에 해당하며, 순차적으로 시행되기 때문에 할당문의 순서가 실행 결과에 영향을 미친다.

**2.**

Blocking과 nonblocking 모두 절차형 할당문의 한 종류이다.

**2-1) Blocking**

Blocking은 할당 기호로써 ‘=’을 사용한다. Nonblocking과의 가장 큰 차이는 begin에서부터 end까지의 코드가 한 줄 씩 순차적으로 이루어진다는 것이며 그 즉시 계산 및 저장이 이루어진다.

아래는 blocking 구문을 사용한 시뮬레이션 코드와 그에 대한 결과이다.

**텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명**스크린샷이(가) 표시된 사진

자동 생성된 설명

매 10ns 마다 z의 값이 변환되며, z가 1일 때 blocking 구문이 실행되는 것을 알 수 있다. 이때 초기에 x = 0, y = 1의 값을 갖고 있는데 x = y, y = x의 코드가 수행 즉시 그 값을 저장하는 blocking 구문이므로 결과적으로 두 값 모두 1이 되는 것을 알 수 있다.

**2-2) Nonblocking**

Nonblocking의 경우에는 ‘<=’ 연산자를 사용한다. Nonblocking의 특징은 blocking과 달리 코드들이 동일한 시간에 동시에 실행된다는 점이다. 다시 말해, begin부터 end까지의 모든 계산이 수행된 뒤에 마지막에 한번에 저장되는 방식이므로 코드의 순서와 무관한 결과가 발생한다.

아래는 nonblocking 구문을 사용한 시뮬레이션 코드와 그에 대한 결과이다.

**텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명**

스크린샷이(가) 표시된 사진

자동 생성된 설명

Blocking 예시와 같이 10ns 마다 z의 값이 변환되며, z가 1일 때 blocking 구문이 실행되는 것을 알 수 있다. 그러나 이번에는 x <= y, y <= x의 코드가 수행 후에 값을 할당하는 방식이기 때문에 매 수행마다 x와 y의 값이 서로 뒤바뀌는 것을 알 수 있다.

**3.**

**3-1) for문**

for문은 C언어의 for문과 동일하게 *for(초기값, 조건식, 제어변수 변화)*의 형식으로 이루어진다. 최초에 초기값을 설정하고 조건식을 확인한 후 제어변수를 변화시키는 식을 수행한다. 후에 조건식 확인과 제어변수 변화 식 수행이 반복된다. C언어와의 차이점은 verilog에서는 always문과 initial문 내에서만 for문을 사용할 수 있다는 점이다.

**3-2) if문**

*if(조건문) else*의 형태를 띈다는 점에서 verilog와 C모두 동일하다. 그러나 verilog의 경우 C에서의 블록을 표시하는 ‘{‘의 역할을 begin과 end가 대신한다. 또한 verilog는 always 블록 안에서만 조건문의 사용이 가능하다. Verilog 역시 C와 마찬가지로 중첩된 조건문을 사용할 수 있다.

**3-3) while문**

if문과 마찬가지로 while(조건문)의 형태를 띄며, C와 동일하게 괄호나의 조건문이 거짓이 되면 반복문을 나가는 형식이다. 또한 if문과 마찬가지로 verilog에선 ‘{‘의 역할을 begin과 end가 수행하며 always와 initial문 내에서만 사용이 가능하다.

**3-4) case문**

|  |
| --- |
| case(조건식)  항\_1 : // 내용  항\_2: begin  //내용  //내용  end  default: //내용  endcase |

case문은 C의 switch문의 역할과 비슷한 역할을 한다. case문은 case, endcase, default로 구성되며 case(조건식)의 형태로 사용된다. 이 조건식의 값과 일치하는 항이 실행되며 일치하는 항이 없을 경우 default항이 실행된다. 각 항이 여러 줄로 구성된 경우에는 begin, end문을 통해 블록을 표시해야 하며 case문의 마지막에는 항상 endcase를 사용해야한다. C와 달리 따로 break를 해줄 필요는 없다.

**4.**

net자료형은 하드웨어 간의 물리적 연결을 위해 사용되는 자료형이다. 또한 연속 할당문에서만 사용이 되는데, assign문처럼 값들이 연속적으로 유지된다. 즉, 만약 연속할당문에서 우변의 값에 변화가 생기면 자동으로 net 자료형의 객체의 값도 변화하게 된다.

net 자료형에는 다양한 종류가 존재한다. wire는 가장 일반적으로 사용되는 net 자료형 중 하나로, 연속할당문을 통해 값을 연결하는데 사용된다. trireg는 wire와 비슷하지만 회로에 정의된 값에 따라 unknown 상태를 의미하는 x 상태를 가질 수 있다는 특징이 있다. wor과 trior은 OR, wand와 triand는 AND 연산을 하는 wire이며 네 종류 모두 다중 구동자를 갖는다. supply0과 1은 각각 회로접지와 전원에 연결될 수 있다. 추가적으로 net 자료형은 default값으로 z를 갖는데 이는 전기적으로 절연됐다는 것을 의미한다. 앞서 말했듯 trireg 자료형의 경우 default값이 x로 논리값 0과 1이 충돌하여 논리값을 확정할 수 없는 unknown 상태를 가진다.