2주차 결과보고서

전공: 수학과 학년: 4학년 학번: 20171273 이름: 심현우

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

Verilog에서 할당문은 말 그대로 객체에 값을 할당해주기 위한 코드이며 질문과 같이 연속, 절차형으로 나뉜다. 둘의 큰 차이점은 값을 할당해주는 객체의 자료형이 다르다는 것이다. 연속 할당문은 net형 객체에, 절차형 할당문은 variable형 객체에 값을 할당한다. 연속 할당문의 경우 right-hand side의 표현 및 값이 변함에 따라 결과를 좌변의 net형 객체에 할당한다. 연속 할당문 암시적 연속 할당과, 명시적 연속 할당이 있는데, 암시적 연속 할당에 경우 net선언과 연속할당을 하나의 statement로 결합하는 것을 의미하고, 명시적 할당은 두개의 statement로 하나는 net을 선언하고 다른 하나는 계속해서 값을 할당하는 역할을 한다. 또한 연속 할당문에는 2가지 지연, 정규지연, net지연이 있다.

보통 연속 할당문은 다음과 같이 표현한다. Assign A = b & c; 여기서 =이 값을 할당해주는 역할을 하기 때문에 할당 연산자라고 하며 위의 표현은 연산자 왼쪽의 A가 오른쪽의 b & c와 연속적으로 구동되기 때문에 연속 할당문이다. 여기서 지연을 추가하게 되면 다음과 같다. Assign #5 A = b & c; 이 표현의 경우 위의 A = b & c와 다른 점은 값이 5시간 단위로 바뀌는 것이다. 이때 #은 할당문에 지연을 지정하는 것이기에 지연 연산자라고 하며 위의 구문에서 #5는 정규지연을 의미한다. 이 정규지연을 관성지연이라고 부르기도 한다. 다른 지연으로 net지연은 할당문에 지연 연산자를 작성하는 것이 아닌 wire #5 A; assign A = b & c; 처럼 작성한다. 이는 net객체에 직접 지연을 거는 것으로 모든 A의 구동에 지연이 발생한다.

절차적 할당문은 위의 연속 할당문과 다르게 초기, 항상, 작업 등 다양한 절차 내부에서 사용되며, 주로 register data type과 memory type을 업데이트 하는 데 사용된다. 절차형 할당문에는 blocking과 non-blocking assignment가 있다. 변수의 값을 할당하는 것에 함수가 사용되며 절차 할당에서 다음 할당을 만나기 전까지 변수는 값을 유지한다. 절차 할당문은 simulation 동안 작동하며 이것이 위의 연속 할당문과 큰 차이점이다. 위에서 언급했듯이 연속 할당문은 단순히 right-hand side에 의하여 할당이 이루어지지만, 절차형 할당문의 경우 simulation이 진행되는 동안 statement와 만날 시 작동하며 이는 프로그램에 영향을 준다.

1. **Blocking 및 nonBlocking문법의 차이를 simulation을 통해 설명하시오.**

Blocking assignments의 경우 blocking을 의미하는 =을 쓰며 blocking의 의미 그대로 다음 실행을 차단한다. 이 다음 실행을 차단하는 것은 다른 의미로 실행에 순서를 부여하는 것을 의미한다. 이는 위의 구문부터 차례대로 구문을 읽는 것으로 생각하면 된다. 다음과 같은 구문을 예를 들어 보자. Initial begin a = 0; b = 1; c = 2; end, always @(posedge clk) begin a = b; c = a; 하게 되면 blocking assignments에 의하여 초기 값은 a가 0, b가 1, c가 2가 된다. 이후 always 구문 내부를 line by line으로 읽으며 a=b;와 c=a가 된다. 따라서 초기 이후 바로 a, b, c값은 a = b = c = 1이 된다.

Non-blocking의 경우 non-blocking을 의미하는 <=을 쓰며 blocking과 다르게 구문을 한줄 한줄 읽는 것이 아닌 동시에 구문을 읽고 실행한다. 이는 non-blocking 말 그대로 차단이 없는 것을 의미하며 blocking과 다르게 순서 또한 없는 것을 의미한다. 위의 blocking 코드에서 blocking을 non-blocking으로 바꾸어 예시를 들어 보자. 초기 initial을 동일하게 a=0, b=1, c=2라고 하고 always 내부 구문을 non-blocking으로 a <= b; c <= a;라 해보자. 위의 blocking의 경우 우선 a에 b의 값을 대입하여 a가 1이 되었고 이후 c에 a값을 대입하여 c또한 1이 되었다. 하지만 non-blocking에서는 동시에 진행되므로 a에는 b의 값을 대입하여 a=1, b=1이 되고 동시에 c에는 initial a값을 대입하여 c는 0이 된다. 따라서 simulation 실행 시 initial에서는 a=0,b=1,c=2이고 이후 a=1,b=1,c=0이 된다.

1. **Verilog의 for문, if문, while문, case문을 C언어와 비교하여 설명하시오.**

Verilog의 for문은 C언어의 for loop와 비슷하다. C언어에서의 for loop는 for문의 조건이 참일 경우 for loop의 내부 코드를 반복하여 실행하고 for의 조건이 참이 아니게 될 경우 loop에서 빠져나온다. Verilog의 경우도 코드 블록을 for loop를 사용하여 고정된 횟수만큼 반복하기 위해 사용한다. for문을 사용하는 기본적인 이유는 동일하지만 문법에서 차이가 있다. C언어의 문법과 verilog의 문법은 다르기 때문에 코드 작성시 주의해야 한다. verilog에서 for문을 사용할 때 조건을 표시함에 있어 C언어에서는 ++연산자를 사용하여 loop를 돌리지만 verilog에서는 ++연산자가 문법에 없기 때문에 이를 풀어서 쓴 i=i+1로 선언해야 한다. 또한 for loop의 시작과 끝을 begin과 end로 나타내야 한다. 예시를 들어 기본적으로 어떻게 사용하는 지 보자. For ( ; ; ) begin (loop code) end 형태로 작성하며 ; ; 에는 C언어와 동일하게 조건이 들어간다. Begin과 end 안에 반복될 코드를 작성한다. While문 또한 veilog에서 코드를 반복하여 실행할 때 쓰고 위의 for문과 같이 문법적으로 다른 점을 제외하면 C언어와 기본적인 쓰임이 비슷하다. while문도 예시를 들어 보면 while <condition> begin (loop code) end 형태로 작성한다.

C언어에서 if문의 경우 다양한 조건에 대해 다른 코드를 작성할 때 쓴다. 이는 verilog에서도 비슷한 이유로 사용된다. 주로 always 구문 안에 사용한다. 위의 for와 while처럼 문법적으로 다른 부분이 있다. If <condition> begin (code) end의 형식으로 작성한다.

case문의 경우 C언어에서는 주로 switch라고 불리기도 한다. C언어에서 switch는 if문과 그 쓰임새가 비슷하다. 여러가지의 case에 따라 다른 행동을 주고 싶을 때 switch문을 사용한다. Switch (변수) { case 변수 값 : }을 통해 변수 값에 해당하는 case마다 다른 코드를 실행한다. Verilog의 case문의 경우 C언어와는 다른 부분이 있다. Verilog의 case문에는 case, endcase와 default가 존재한다. Case와 endcase는 C언어와는 다른 문법적인 부분이고 default는 C언어에서 존재하지 않는 구성이다. 우선 case를 통해 case문의 시작을 알린다. 이는 C언어의 Switch와 같은 역할이다. 이후는 C언어와 비슷한 방법으로 case별 코드를 작성한다. 마지막으로 default의 코드를 작성하는데 이는 위의 case와 부합하지 않을 시 default의 코드를 실행하는 것이다. C언어에서는 따로 작성해주지 않는 이상 이 default를 굳이 작성하지 않아도 된다. 따라서 verilog에서 default가 C언어와 다른 가장 큰 부분이다. 이후 endcase로 구문의 끝을 알린다. Verilog의 case문에서 case item, 즉 다른 case들을 나타내는 부분에서 casez와 casex가 존재하는데 이도 C언어와 다른 부분이다. Casez와 casex란 그 case의 요소를 무시하고 조건을 판단하는데 쓰인다. Casez의 경우 z요소를 무시하고 casex의 경우 z와 x 모두 무시하고 판단한다.

1. **Verilog의 net형 자료형에 대해서 조사하시오.**

Verilog의 자료형은 net형과 variable형 두가지가 있다. Net형 자료형은 구조를 구성함에 있어 model connections을 하기 위해, 즉 물리적인 연결을 추상화하고 구체적으로 나타내기 위해 사용된다. Verilog의 코드를 작성함에 있어 예시를 들면 wire나 tri, trior, wand 등이 있다. 이 중 wire가 가장 자주 사용되는 유형이다.