

Verilog 기본 문법

(Vivado Verilog Grammar)

Verilog 기본 문법

- Verilog 기본 단위
 - Head
 - Declaration
 - Body
- Verilog - Body 구조 분석
 - 순차 회로
 - 조합 회로
 - 조건 문
 - 루프 문

Verilog 기본 문법

1. Verilog 기본 단위

- Head

- Module 이름과 Port 목록들을 지정
- 선언 방법
 - `module Module_name (port_list);`
 - Declaration
 - Body
 - `endmodule`

<머리부> `module module_name (port_list) ;`

<선언부> `port 선언`

`reg 선언`

`wire 선언`

`parameter 선언`

<몸체부> 하위모듈 인스턴스

게이트 프리미티브

`always` 문, `initial` 문

`assign` 문

`function`, `task` 정의

`function`, `task` 호출

`endmodule`

Verilog 기본 문법

1. Verilog 기본 단위

- Declaration

- 포트 목록에 나열된 포트들의 방향, 비트 폭, 자료형 및 Parameter 선언 등 모듈에서 필요로 하는것들을 선언
- 선언 방법
 - `module Module_name (port_list);`
 - **Declaration**
 - Wire : 회로를 연결하는 모든 선 > 데이터 전달가능 / 저장 X
 - Body 부에서 선언 시 assign 문으로 값을 인가 가능
 - Ex)
 - `wire c_out; // 1비트 wire`
 - `wire [7:0] data; // 8비트 wire`
 - `wire msb = data[7] // data[7]을 msb로 재정의`
 - Reg : 데이터를 저장 가능하며, 레지스터에 새로운 값이 들어오기 전까지는 데이터를 유지
 - Body 부에서 선언 시 always 문으로 값을 인가 시 레지스터로 선언해야 함
 - Ex)
 - `reg_sum; // 1비트 레지스터`
 - `Reg [7:0] bus; // 8비트 레지스터`
 - Parameter : 모듈 내에서 사용하는 상수를 정의할 수 있으며, 합성 또는 시뮬레이션 시에 값이 치환되며, 많이 사용되는 상수의 경우 숫자를 직접 사용하는 것보다 파라미터를 이용하는 것이 효율적임
 - 하위 모듈의 파라미터를 바꿀 경우 defparam을 이용하여 바꿀 수 있음
 - Ex)
 - `parameter SIZE = 16; // 정수로 파라미터 정의`
 - `reg [SIZE-1:0] A_BUS; // 파라미터 값을 이용`
 - `Parameter S_IDLE = 4'b000; // 4비트로 파라미터 정의`
 - Body
 - `endmodule`

Verilog 기본 문법

1. Verilog 기본 단위

- Body

- 회로의 기능, 동작, 구조 등을 표현하는 다양한 베릴로그 구문들로 구성되며, 사용되는 구문들은 논리 합성용 구문, 시뮬레이션용 구문, 라이브러리 설계용 구문으로 구분
- 선언 방법
 - module Module_name (port_list);
 - Declaration
 - **Body**
 - assign 문
 - Always 문
 - If ~else 문
 - Case 문
 - For 문
 - endmodule

Verilog 기본 문법

2. Verilog – Body 구조 분석

- 순차 회로 (Sequential Logic)

- 입력 뿐만 아니라 현재 상태에 따라 값이 다르게 나올 수 있는 회로
- 현재 상태를 기억하고 있기 때문에 메모리 소자(Latch 또는 Flip-Flop)를 가지고 있음
- always문의 타이밍 제어가 이벤트일 경우 Sensitivity List에 해당하는 이벤트가 발생할 경우 아래 순차회로가 실행됨 (이벤트가 발생하지 않을 경우 값을 유지)

• Always 문

- 값을 할당할 수 있는 데이터 형은 레지스터 형이며, 좌변에는 레지스터 형만 올 수 있으며 생략 할 수 없음
- 우변에는 레지스터, Wire, Parameter 형을 모두 사용할 수 있으며 산술·논리·조건 연산식을 사용할 수 있음
- Ex)

• Always @ (posedge clk or negedge reset) // @는 타이밍 제어, 그 뒤는 sensitivity list

• Begin // 순차회로 기술

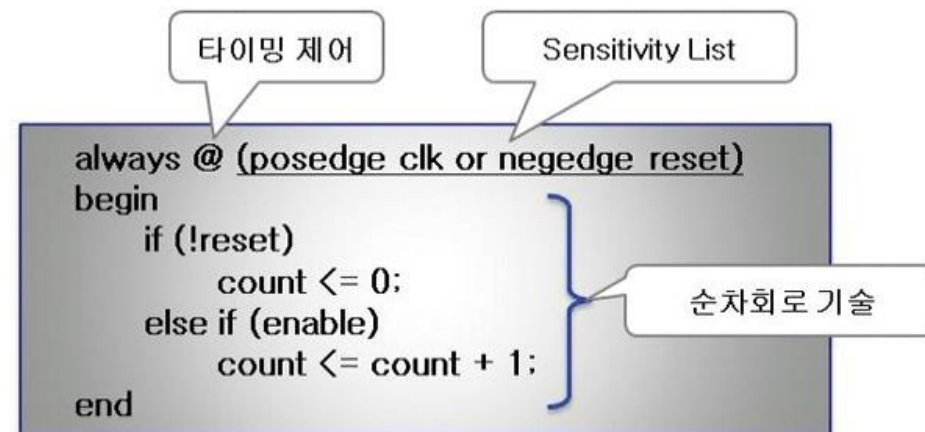
• If (!reset)

• Count <= 0;

• Else if (enable)

• Count <= count + 1; // 순차회로 기술 end

• end

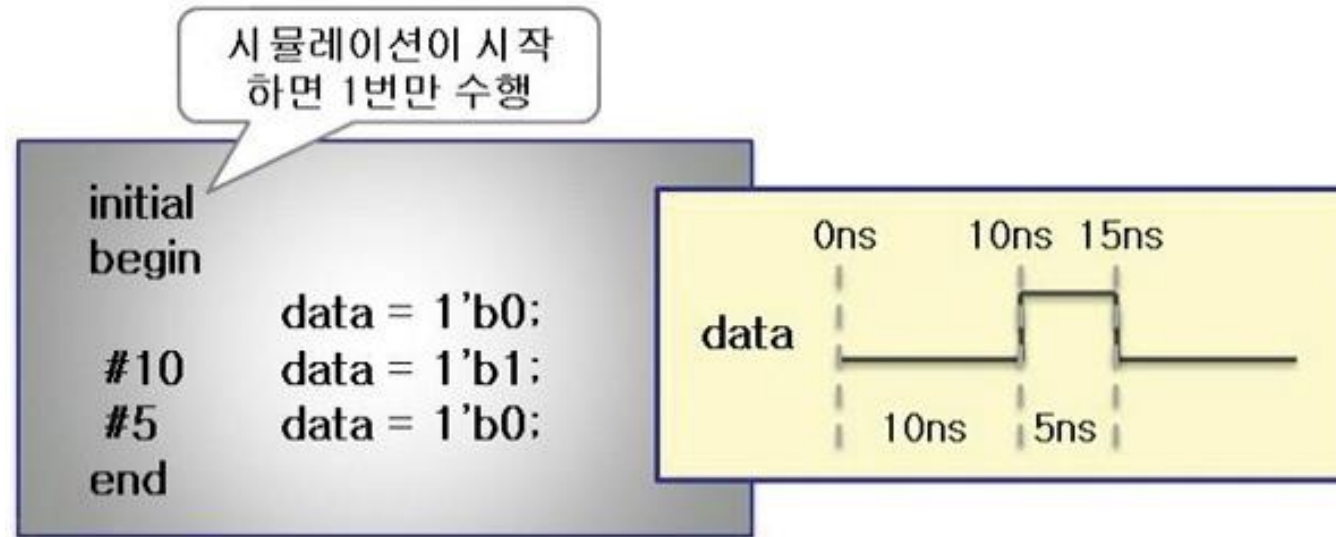


Verilog 기본 문법

2. Verilog – Body 구조 분석

- **Initial 문**

- **시뮬레이션**을 위한 구문으로 순차적으로 신호를 인가할 때 사용
- 시뮬레이션이 시작하면 모든 Initial 구문이 실행되어 파형을 만들며, 시간 지연을 위해서 일반적으로 블로킹 구문을 사용하며 타이밍 제어를 진행
- 기준시간이 1ns라고 가정 하면 시뮬레이션 시작과 동시에 data에는 1'b0 이 할당
- 10ns 시간 지연 후에 1'b1 이 할당되며 다시 5ns 시간 지연 후에 1'b0 이 할당됨
- 마지막 1'b0 은 15ns 시간에 할당되게 됨
- Ex)

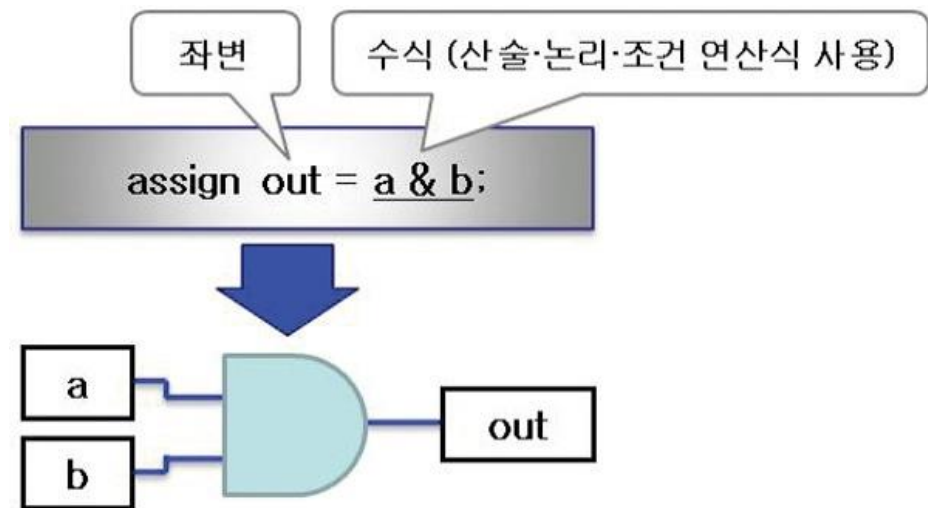


Verilog 기본 문법

2. Verilog – Body 구조 분석

- 조합 회로 (Combinational Logic)

- 입력에 따라 출력 값이 정해져 있는 회로로 Latch 또는 Flip-Flop와 같이 기억 소자가 없는 회로
- 실제 구현될 때 AND, OR, NOT 게이트로 구현되는 회로
- **Assign 문**
 - assign 문의 좌변에는 wire로 선언된 데이터 형만 올 수 있으며 선언이 생략되어 있는 경우 1비트 wire로 인식하게 됨
 - 우변에는 레지스터, Wire, Parameter 형을 모두 사용할 수 있으며 산술·논리·조건 연산식을 사용할 수 있음
 - Ex)
 - Assign out = a + b; // 산술 연산
 - Assign all_one_flag = &data[7:0]; // 리덕션 연산
 - Assign out = sel ? a : b; // 조건 연산

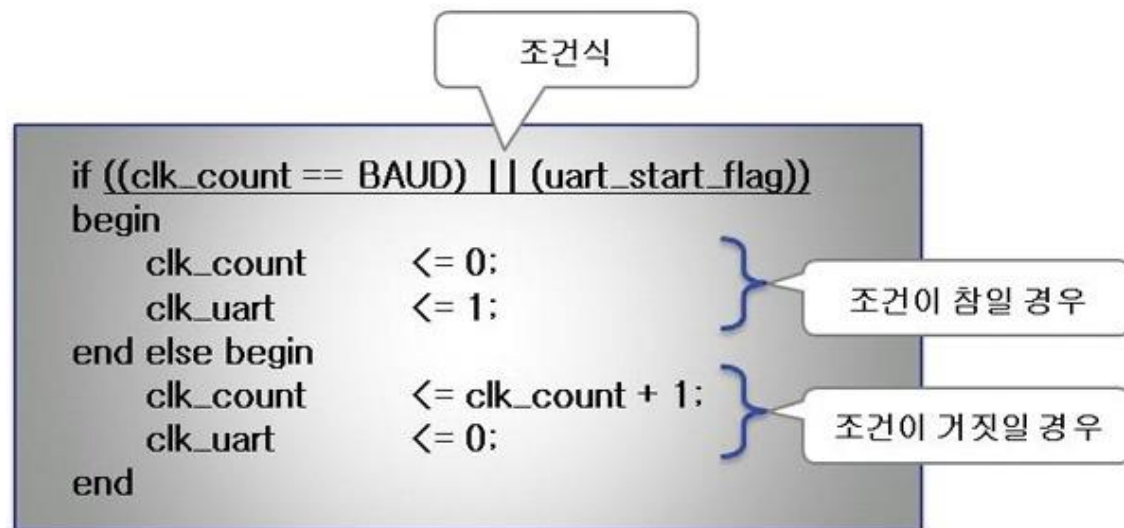


Verilog 기본 문법

2. Verilog – Body 구조 분석

- 조건 문

- 조건문은 기술된 조건에 따라 다른 문장을 실행할 때 사용
- 조건문을 중복하여 사용할 경우에는 실행 범위를 begin-end로 정확하게 명시하는 것이 좋음
- 범위가 명확하지 않은 경우 오동작을 할 수 있음
- 조건식 내에는 다양한 연산자를 사용할 수 있음
- **If-else 문**
 - 기본적인 형태는 If-else 형태로 C언어와 같은 방법으로 사용할 수 있음
 - If 문만 단독으로 사용하거나 If - else if와 같은 형태 또는 조건문을 중복하여 사용할 수도 있음
 - Ex)

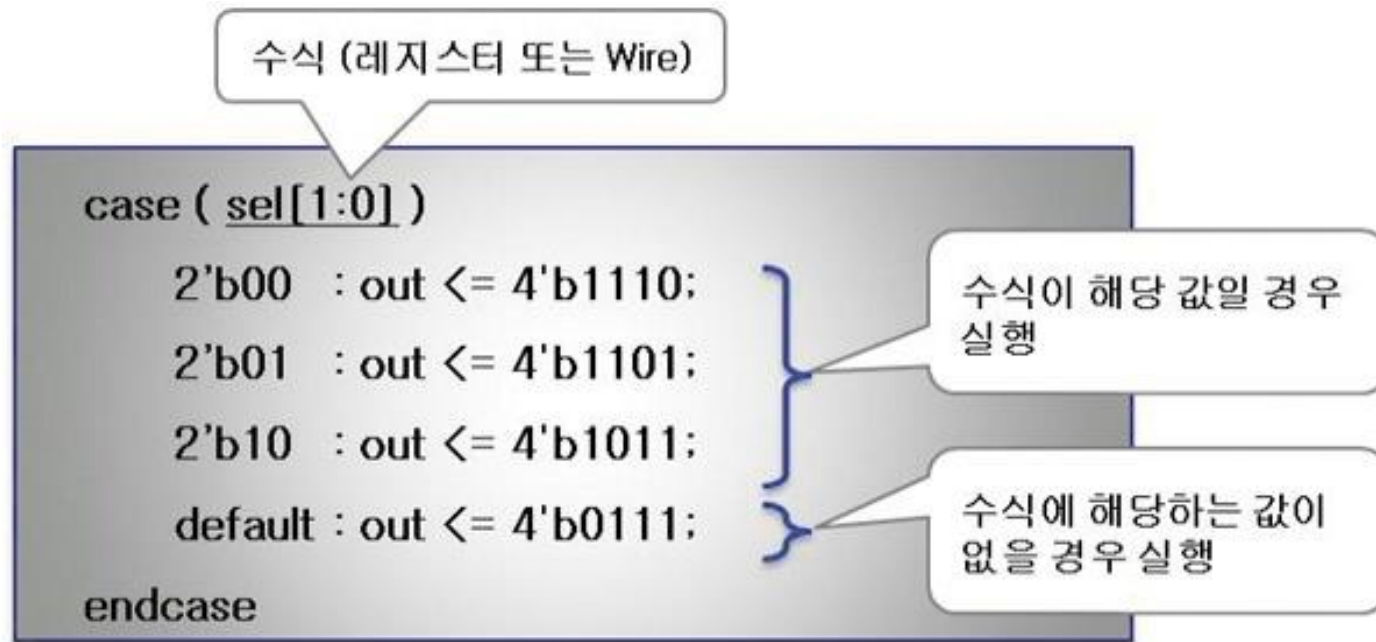


Verilog 기본 문법

2. Verilog – Body 구조 분석

- **case 문**

- 조건문을 사용할 때 조건이 많은 경우에는 If-else 문보다 Case 문을 사용하는 것이 효율적임
- Case 문은 C언어의 switch-case 구문과 같은 동작을 진행
- Case 문의 마지막에는 "endcase"를 사용하여 case close 명시
- Ex)

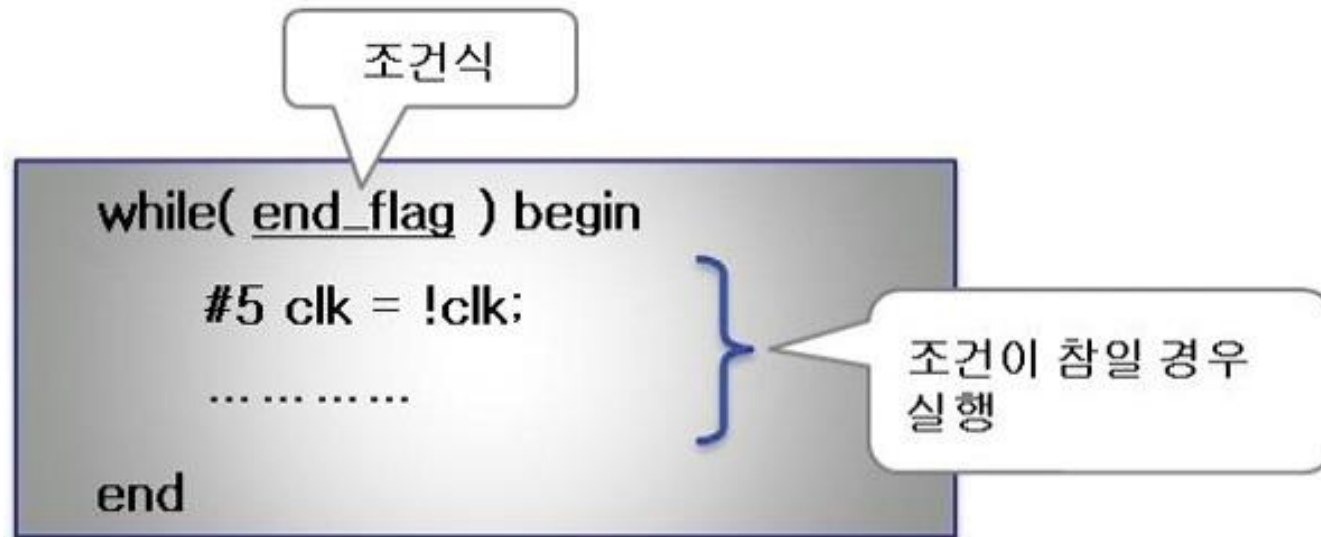


Verilog 기본 문법

2. Verilog – Body 구조 분석

- 루프 문

- 루프문은 일반적으로 **시뮬레이션**을 위한 TestBench 작성 시에만 사용
- 제한된 형태의 for문 같은 경우 합성이 되기도 하지만 합성툴에 따라 다르기 때문에 design source file 에는 되도록이면 사용하지 않는 것이 좋음
- **While 문**
 - While 문은 조건식이 참일 경우 실행
 - 루프 실행 횟수가 조건에 따라 다르기 때문에 합성이 불가능
 - Ex)

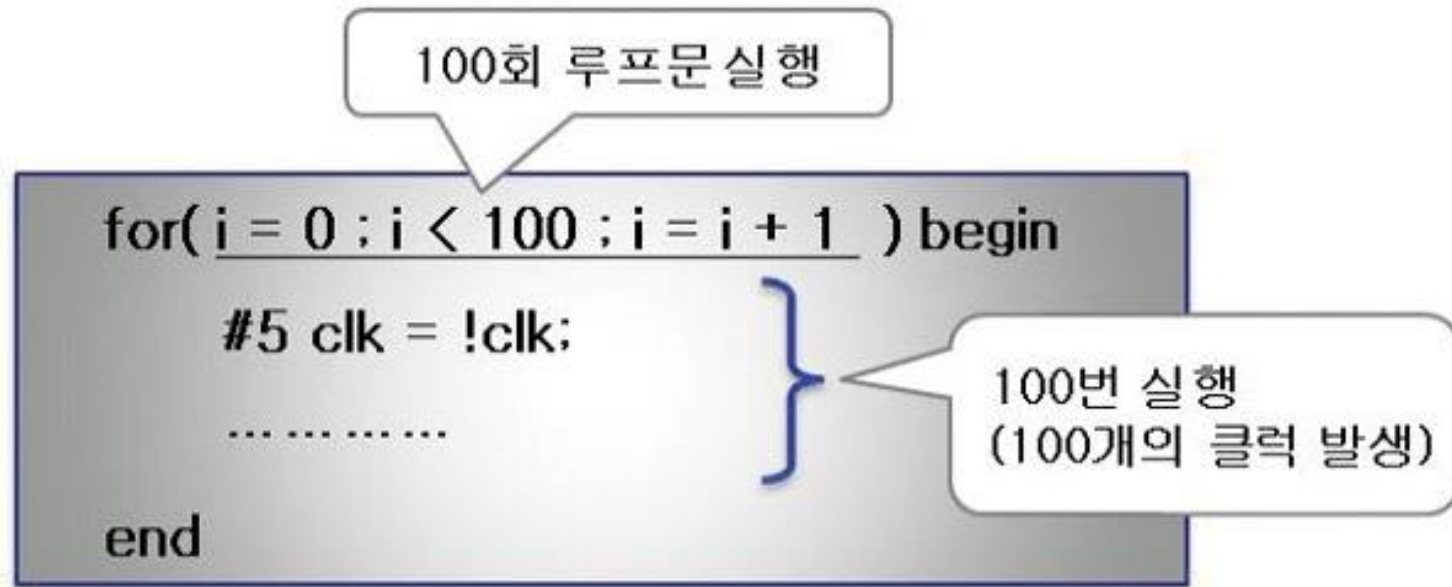


Verilog 기본 문법

2. Verilog – Body 구조 분석

- **for 문**

- For 문은 루프의 실행 횟수가 정해진 경우 사용
- 실행 횟수가 정해지지 않은 경우 while 문을 사용하는 것이 좋음
- 합성툴에 따라 제한된 횟수의 for 문은 합성 가능한 경우가 존재
- Ex)

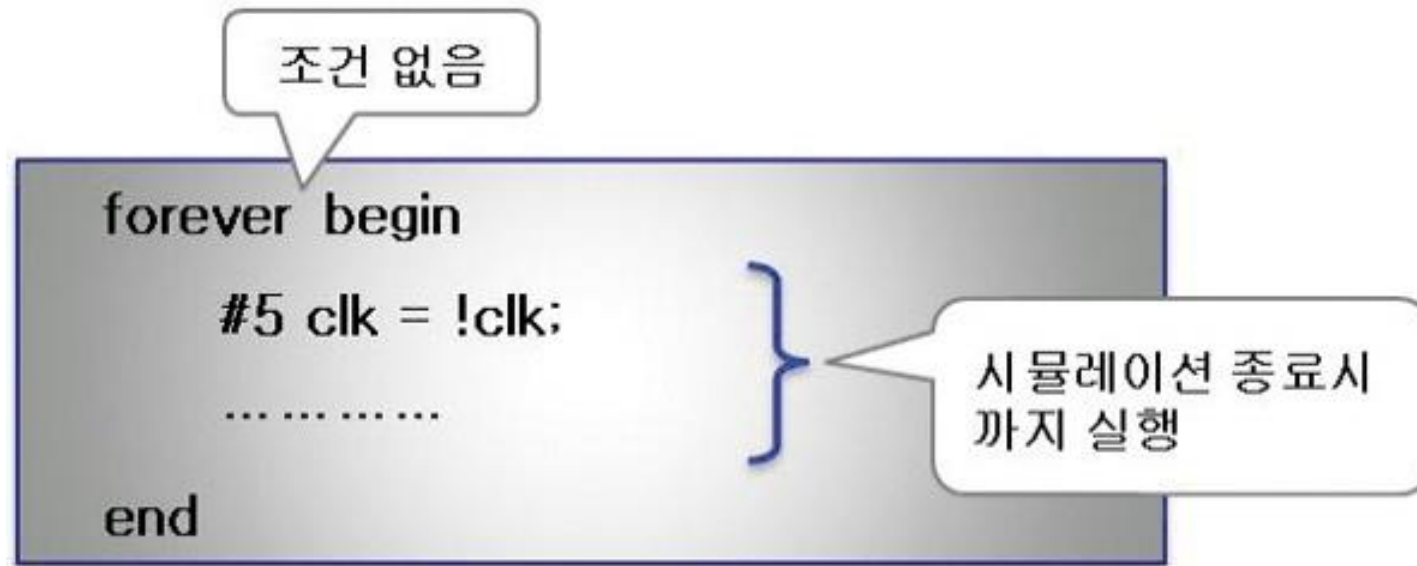


Verilog 기본 문법

2. Verilog – Body 구조 분석

- **Forever 문**

- Forever 문은 실행문을 무한히 반복할 때 사용
- disable 문장을 사용하여 종료할 수 있지만 일반적으로 시뮬레이션이 종료될 때까지 동작시킬 때 사용
- 시뮬레이션 종료는 \$finish task
- Ex)



Verilog 기본 문법

2. Verilog – Body 구조 분석

- **Repeat 문**

- Repeat 문은 특정 횟수만큼 실행할 때 사용
- 횟수를 나타내는 문장에는 상수 또는 변수, 파라미터가 올 수 있음
- 레지스터와 같은 변수가 사용되더라도 Repeat 문이 처음 실행할 때 값 만큼 실행
- 루프 실행 중에 레지스터의 값이 변경되더라도 루프 실행 횟수에 영향을 주지 않음
- Ex)

