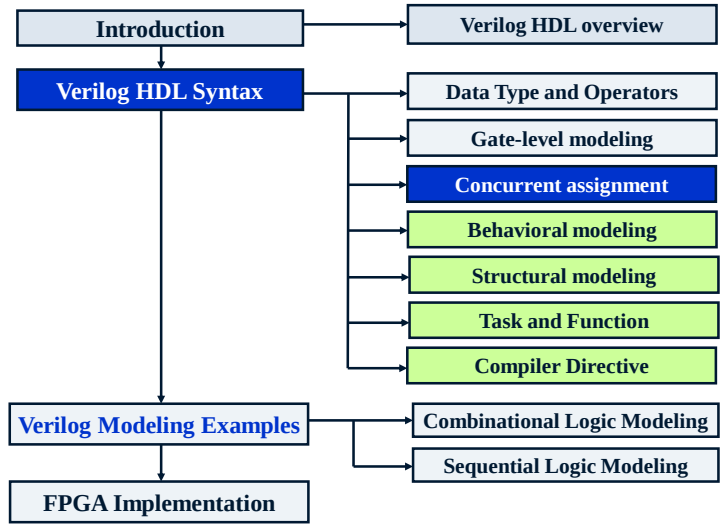
할당문

Kyung-Wook Shin
kwshin@kumoh.ac.kr

School of Electronic Eng.,
Kumoh National Institute of Technology

Learning Map



4.0 할당문

- 할당문
  - 연속 할당문 : net형 객체에 값을 할당
    - assign 문

```
continuous_assign ::= assign [driving_strength] [delay] net_assignments;
```

  - 절차형 할당문 : variable형 객체에 값을 할당
    - always 블록, initial 블록, task, function 내부의 할당문
    - blocking 할당문 : 할당기호 = 을 사용
    - nonblocking 할당문 : 할당기호 <= 을 사용
  - 절차형 연속 할당문 (Procedural Continuous Assignment; PCA)
    - assign - deassign 문, force - release 문

4.1 연속 할당문

- 연속 할당문
  - assign 문을 이용하여 net형 객체에 값을 할당
  - 우변의 값에 변화(event)가 발생했을 때 좌변의 객체에 값의 할당이 일어남
  - 단순한 논리 표현(연산자 사용)을 이용한 조합논리회로 모델링에 이용
- 함축적(implicit) 연속 할당문
  - net 선언문에 연속 할당문을 포함시킨 경우

예 4.1.1 함축적 연속 할당문

```
wire  mynet = enable & data;
```

```
wire mynet;

assign mynet = enable & data;
```

함축적 연속 할당문과 등가

4.1 연속 할당문


```
assign na = ~(in1 & in2);           // 2 input NAND
assign out = (sel==1) ? d1 : d0;    // 2-to-1 MUX
assign carry = (cnt10==4'h9);
assign sum = a + b;                 // 덧셈 회로
```

예 4.1.2 연속 할당문을 이용한 4비트 가산기

```
module adder (sum_out, carry_out, carry_in, ina, inb);
   output [3:0] sum_out;
   output       carry_out;
   input  [3:0] ina, inb;
   input        carry_in;

   assign {carry_out, sum_out} = ina + inb + carry_in;
endmodule
```

코드 4.1

4.1 연속 할당문


```
module select_bus (busout, bus0, bus1, bus2, bus3, enable, s);
   parameter n = 16;
   parameter Zee = 16'bz;
   output [1:n] busout;
   input  [1:n] bus0, bus1, bus2, bus3;
   input        enable;
   input  [1:2] s;
   tri    [1:n] data;  // tri net declaration

// 연속 할당을 갖는 net 선언(함축적 연속 할당문)
   tri [1:n] busout = enable ? data : Zee;

// 4개의 assign 문을 이용한 표현
   assign data = (s == 0) ? bus0 : Zee;
   assign data = (s == 1) ? bus1 : Zee;
   assign data = (s == 2) ? bus2 : Zee;
   assign data = (s == 3) ? bus3 : Zee;

/* 하나의 assign 문으로 4개의 연속 할당을 표현(위의 4개의 assign 문과 등가임)
   assign data = (s == 0) ? bus0 : Zee,
          data = (s == 1) ? bus1 : Zee,
          data = (s == 2) ? bus2 : Zee,
          data = (s == 3) ? bus3 : Zee; */
endmodule
```

예 4.1.3 연속 할당문을 이용한 16비트 출력 버스

코드 4.2

4.1.1 할당 지연과 net 지연

- 연속 할당문의 지연값 지정
  - assign 뒤에 지연 연산자(#)를 사용하여 지정
  - 우변 피연산자 값의 변화에서부터 그 값이 좌변에 할당되기까지의 시간 간격을 지정

```
assign  #10 wireA = a & b;
```

```
wire    #10 wireA = a & b;  // 함축적 연속 할당문의 지연
```

- net 지연
  - net 선언문에서 지연 값을 지정
    - 지정된 net 지연이 경과한 후에 할당이 이루어 짐
    - 해당 net를 구동하는 모든 구동자 (게이트 프리미티브, 연속 할당문 등)에 영향을 미침

```
wire  #10 wireA;
```


4.1.1 할당 지연과 net 지연

- 관성지연 (inertial delay)
  - Verilog HDL의 default 지연
  - 지정된 지연 값보다 입력 신호의 변화 폭이 작은 경우, 입력 신호의 변화가 출력에 영향을 미치지 않음
  - 관성지연은 게이트 수준 모델링에도 동일하게 적용
  - 관성지연의 적용 과정
    ① 우변 수식의 값이 평가된다.
    ② 평가된 우변의 값이 좌변에 할당 예정된 값과 다르면, 현재 예정된 할당 event가 취소된다.
    ③ 새로운 우변의 값이 좌변의 현재 값과 동일하면 할당을 위한 event가 예정되지 않는다.
    ④ 새로운 우변의 값이 좌변의 현재 값과 다르면, 지연이 계산되고 주어진 지연 후에 새로운 할당 event가 일어나도록 예정된다.

4.1.1 할당 지연과 net 지연

예 4.1.4 연속 할당문에서의 관성지연

```
module inertial_delay();
  reg a, b;

  assign #30 wireA = a & b;
  initial begin
         a = 1'b0;
         b = 1'b0;
    #50  a = 1'b1;
         b = 1'b1;
    #50  a = 1'b0;
         b = 1'b0;
    #50  a = 1'b1;
         b = 1'b1;
    #20  b = 1'b0;
    #50;
  end
endmodule
```

코드 4.3

4.1.1 할당 지연과 net 지연

예 4.1.4 연속 할당문에서의 관성지연

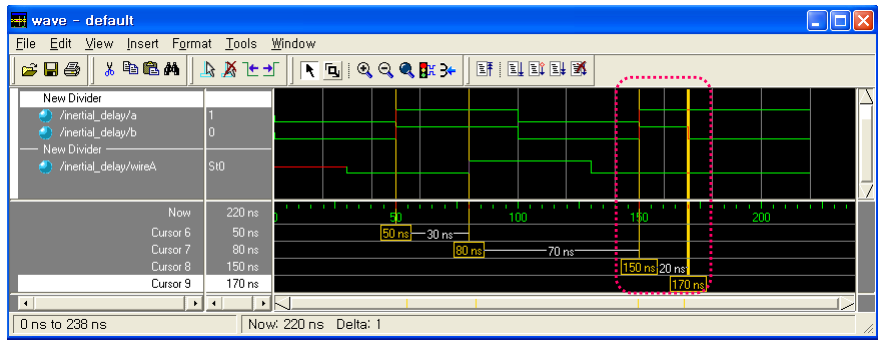


그림 4.2 코드 4.3의 시뮬레이션 결과

4.2 절차형 할당문

- 절차형 할당문
  - reg, integer, real, time, realtime 등 variable에 값을 갱신
  - 지연을 갖지 않으며, 다음 절차형 할당문에 의해 값이 갱신될 때까지 변수에 할당된 값을 유지
  - always, initial, task, function 등의 프로시저(procedure) 내부에서 사용
  - **문장의 실행**에 의해 좌변 variable에 값이 할당되는 소프트웨어적인 특성
    - 우변 수식의 event 발생과는 무관
    - 할당문들의 순서가 시뮬레이션 결과에 영향을 미칠 수 있음
- 함축적 변수 할당문
  - variable 선언문에서 variable에 대한 초기 값을 설정
  - 배열에 대한 함축적 변수 할당은 허용되지 않음

4.2 절차형 할당문

예 4.2.1 절차형 할당문

```
module proc_assignment(clk, a, b, out);
  input  clk, a, b;
  output out;
  reg    out, c;

  always @(posedge clk) begin
    c = a & b;       // blocking assignment
    out <= c;        // nonblocking assignment
  end
endmodule
```

코드 4.4

```
reg[3:0] areg = 4'h4;
```

```
reg[3:0] areg;

initial areg = 4'h4;
```

함축적 변수 할당문과 등가

```
reg [3:0] array [7:0] = 0; //illegal (배열에 대한 함축적 변수 할당문)
```