Simulation program (Test bench)

◉ Simulation program은 test bench라고도 하며, 구현한 H/W를 검증하기 위한 프로그램.

◉ Test bench의 구조 <그림1>

module test_bench;
reg 선언 // 입력신호 wire 선언 // 출력신호 local 변수 선언
Simulation model 호출
Test 입력 program : initial 문 사용하여 검증 시작 및 task 호출 등 표시 : 동작 관측
endmodule

a) 입출력 및 변수를 선언 : test bench에서 구현한 모듈의 입력을 reg, 출력을 wire로 선언해 주어야 함.

b) 구현한 H/W를 호출(instantiation).

c) System task를 사용하여 test vector를 입력하며 동작을 확인.

- $stop : simulation을 중단하고, 대화(interactive)모드가 됨.
- $finish : simulation의 완전한 종료.
- $monitor : 한번 호출되면 계속해서 실행되며, 표시해야 할 신호에 변화가 있을 때만 출력(initial문 중에 1회 정의로 충분)
- $display : 호출될 때만 실행되며, 신호의 변화가 없더라도 출력(initial문에서 반복 호출 필요). C언어의 printf와 유사. $fdisplay <=> fprintf와 유사.
- #number : number만큼의 simulation time 만큼 지연.

예1) 4X1 MUX의 source code와 test bench

```
`define MSB 4                                    // define 문 : MSB는 4
module mux_4to1 (i0, i1, i2, i3, sel, out);      // module 선언
  input [`MSB-1:0] i0, i1, i2, i3;               // 4bit input
  input [1:0] sel;
  output [`MSB-1:0] out;
  reg [`MSB-1:0] out;
  always @(sel or i0 or i1 or i2 or i3) // always 안에서 입력으로 사용되는 모든
    begin                                        // 변수가 sensitive lists에 있어야 함.
    if(sel==0)       out = i0;  //sel이 0이면,
    else if(sel==1) out = i1;  //sel이 1이면,
    else if(sel==2) out = i2;  //sel이 2이면,
    else out = i3;
    end
endmodule
```

- `(back tick)define <macro_name> number
  => C의 #define과 유사. `<macro_name>으로 사용된 구문을 만날 때마다 원래의 macro로 교체.

- 4X1 MUX의 test bench

```
module mux_4to1_tb;
reg [3:0] i0,i1,i2,i3; // 데이터의 입력
reg [1:0] sel;         // 데이터의 입력
wire [3:0] out;        // 데이터의 출력
mux_4to1 mux(i0, i1, i2, i3, sel, out); //mux_4to1의 instantiation
initial
  begin
  #10 i0 = 4'b0000;  //데이터 입력
      i1 = 4'0001;
      i2 = 4'0010;
      i3 = 4'0011;
      sel = 2'b00;   // i0 선택
  $display("time %d, i0=%b, i1=%b, i2=%b, i3=%b, sel=%b, out=%b ₩n", $time,
  i0, i1, i2, i3, sel, out);
  #10 sel = 2'b01;  // i1 선택
  $display("time %d, i0=%b, i1=%b, i2=%b, i3=%b, sel=%b, out=%b ₩n", $time,
  i0, i1, i2, i3, sel, out);
  #20 sel = 2'b10;  // i2 선택
  $display("time %d, i0=%b, i1=%b, i2=%b, i3=%b, sel=%b, out=%b ₩n", $time,
  i0, i1, i2, i3, sel, out);
  #25 i0 = 4'b1111;
      i1 = 4'b1110;
      i2 = 4'b1100;
      i3 = 4'b1000;
  #5 sel = 2'b11;  // i3 선택
  $display("time %d, i0=%b, i1=%b, i2=%b, i3=%b, sel=%b, out=%b ₩n", $time,
  i0, i1, i2, i3, sel, out);
  #10 $stop;      // simulation 종료
  end
endmodule
```

예2) Counter의 source code와 test bench

```
module CNT_UP_DOWN(Clock, Reset, Up, Down, Count);
  input Clock, Reset, Up, Down;
  output [3:0] Count;
  reg [3:0] Count;
  always @(posedge Clock) begin // Clock의 상승 에지마다
    if(Reset)                   // Reset이면 0으로 초기화
    Count = 0;
    else
      case({Up,Down})  // Concatenation operator를 사용하여 Up, Down -> 2bit
      2'b00 : Count = Count ;      // Up==0, Down==0 --> hold signal
      2'b10 : Count = Count+1 ;  // Up==1, Down==0 --> up counting
      2'b01 : Count = Count-1 ;  // Up==0, Down==1 --> down counting
      default : Count = Count ;   // Up==1, Down==1 --> hold signal
      endcase
  end
endmodule
```

```
module CNT_UP_DOWN_tb;
reg Clock, Reset, Up, Down;
wire [3:0] Count;
CNT_UP_DOWN CNT(Clock, Reset, Up, Down, Count); // CNT_UP_DOWN의
initial                                          // instantiation
begin
  Clock = 1'b1; // Clock의 초기화
  Reset = 1'b1; // Reset의 초기화
end
always
  #5 Clock = ~Clock;  // Clock generation
initial
begin
#20 Up = 0; Down =0;  // select hold
#30 Up = 1; Down =0;  // select up
#30 Up = 0; Down =1;  // select down
#30 Up = 1; Down =1;  // select hold
end
endmodule
```

- Clock generation : 위의 코드에서 Clock는 10 simulation time.
- Simulation time란 시뮬레이션의 단위로 `timescale을 사용하여 설정가능.

예)
```
`timescale 10ns/ns      /* simulation time : 10ns, sampling time : 1ns
                          #1은 10ns이며, #number는 number*10ns가 된다.*/
```