The initial Block

- The other procedural block is called the initial block
- `initial` can use any of the constructs used in `always`
- `initial` blocks are only for implementing testbench code
- `initial` blocks are ignored by synthesis
- `initial` executes only once, but not necessarily first
- `initial` goes dormant after executing its last instruction
- between the `begin` and `end` statement of `initial` execution is in-order sequential
The initial Block

The initial block

- Structure of the initial block:

```plaintext
initial <statements>
//OR
initial begin
  <statement>
  <statement>
  <statement>
end
//OR
//delay_or_event_control could be a delay, an edge, or level
initial <delay_or_event_control> begin
  <statement>
  <statement>
  <statement>
end
```
The initial Block

Initial block execution

- It can execute in zero time, or be timed with delays
- For example,...
module tb();
    parameter CYCLE = 100;

    reg clk, reset_n;

    //create the clock
    initial begin
        clk = 0;
        forever #(CYCLE/2) clk = ~clk;
    end
    //time the release of reset
    initial begin
        reset_n = 0; //initialize reset asserted
        #(3*CYCLE) @(posedge clk); //wait 3 clock cycles and one pos edge
        @(negedge clk); reset_n = 1; //at the next negedge, release reset_n
    end
endmodule
The initial Block

Brief Digression on Delays

- Delays are not synthesizable, use only in testbench code
- In general, the # means delay
- Examples...

// #2 - delay by two time units before executing the next statement
// example:
always wait(en) begin
  #2 out_sig = q_bar; //assign out_sig after 2 time unit delay
end

// #CYCLE - delay by the defined constant delay
// example:
forever #(CYCLE/2) clk = ~clk; //invert clk every half cycle

Note that the delay is specified as time units, not nS or pS.
The initial Block

Digression on Delays (cont.)

- To delay while waiting on a signal *level* we use `wait`

```vhdl
//wait until enable goes true
//executes immediately if enable is true
always wait(enable) begin ...... end
```

- To do some action forever, we use `forever`

```vhdl
//produce a clock until simulation finishes
forever #(CYCLE/2) clk = ~clk; //invert clk every half cycle
```

- To delay while waiting on a *edge* we can use `@(posedge signal)` or `@(negedge signal)`

```vhdl
//wait till a falling edge, then deassert reset_n
 @(negedge clk); reset_n = 1;
```
Digression on Delays (cont.)

- Units and precision for delay values is given by ‘timescale’
- The character is a ”backtick”, not a single quote
- A timescale directive should be placed prior to the use of #
- Modelsim defaults to 1ns precision
- Xilinx and Altera simulators use 1ps precision.

//‘timescale <time_unit> / <time_precision>
//
//example: Use 1ns time units and 1ps precision
‘timescale 1ns/1ps