VERILOG 3:
TIME AND DELAY
4 Realms of Time and Delay

1) Verilog simulation: “wall clock” time
2) Verilog simulation: timing within the simulation
   a) “#” delays discussed in following slides
3) Synthesis output HW: timing analysis
   a) Simple models using “#” delays in library
   b) More sophisticated Static Timing Analysis (end of 180B)
4) Synthesis output HW: actual circuit propagation delays ($t_p$)
Delays

• Delays may be inserted into always and initial blocks to cause the simulator to let “simulation time” advance.

• Syntax:
  - `#n` // delay of n time units
  - Example:

```vhdl
always @(...) begin
  a = b;
  #5;       // 5-unit delay
  a = ~c;
  #5;
  a = (c|d)^{(e|f)};
end
```

![Waveform Diagram](image-url)
Delays

- Delays are normally used in three places:
  1. The testbench Verilog where it is essential
     - Example: to time input signals
     - Example: the clock generator (see Verilog Testing notes)
     - Example code:

```verilog
//Example testbench to generate input signals
always @(.) begin
    reset = 1'b1;
    in    = 16'h0000;
    #10;   // 10-unit delay
    reset = 1'b0;
    in    = 16'h0001;
    #10;
    in    = 16'h0002;
    #10;
    in    = 16'h0003;
    #10;
    ...
end
```
Delays

- Delays are normally used in three places:
  2) In flip-flop declarations in “hardware(!) verilog”
     - To set a clock-to-Q delay for the purpose of increasing waveform readability
     - Usage will normally produce a warning from synthesis tools
     - Details and syntax are given in a later lecture
  3) In gate libraries to provide crude delay estimations
     - We will not work on this in this class
Concurrent Operation

- You should think of verilog modules as operating on independent circuits (remember *hardware* orientation).

```verilog
always begin  // this block executes repeatedly without pausing
    a = (b & c) | d;
    #5;        // 5-unit delay
    a = ~a;
    #5;
end

always begin
    f = ~(g ^ h);
    #7;        // 7-unit delay
    f = ~f;
    #7;
end
```

![Waveforms](image)
Setting the timescale of “#” delays

- `timescale time_unit base / precision base
- The first argument specifies “#1” delay
- The second argument specifies the precision with which delays may be specified
- Base is {s, ms, us, ns, ps, fs}
- Ex: `timescale 1ns/10ps
  - #5 would produce a 5 ns delay