

Simulation

Exercise 1:

1. typical inputs, 4, 7, 25
2. minimum and maximum valid inputs, 0, $2^{15}-1$
3. invalid inputs, and -1
4. randomly-chosen values. 1237

Give examples of appropriate test inputs for each of the above categories if you were testing a circuit that computed the square root of a 16-bit signed number.

Exercise 2: What's the difference between `wait(x) y='1;` and `@(x) y='1;?`

`wait(x)` → waits until $x \neq 0$
`@(x)` → waits until x changes.

Exercise 3: How could you:

- (a) terminate the simulation if a test vector failed?
- (b) change the clock frequency to 10 MHz?
- (c) print each test vector as it's read?
- (d) assert the reset input for two clock cycles?

```

// wait for error or end of file
wait (n < 0) $stop();
end

// 1 MHz clock (2x500ns period)
always #0.5us clk = ~clk;

// check outputs and change inputs on
// each falling clock edge
always @(negedge clk) begin
    if ( out != out_ ) begin
        $display("Error: %d %d %d", in, out, out_);
        n = $fscanf(fd, "%d,%d,%d", reset, in, out_);
    end
end
endmodule

```

Handwritten annotations on the code:

- A line from "(b) #50ns" points to the `#0.5us` delay in the clock generation line.
- A circled "2" points to the `begin` keyword in the `if` statement.
- A circled "1" points to the `$stop()` function call.
- A circled "2" points to the `end` keyword of the `if` block.
- A circled "2" points to the `$display` function call.
- A circled "1" points to the `$fscanf` function call.
- A circled "2" points to the `end` keyword of the `always` block.
- A circled "2" points to the `endmodule` keyword.

Handwritten test vectors:

```

1,0,0 (d)
1,0,0
0,1,1
0,2,1
0,3,4
0,4,4
0,5,8
1,7,0

```

Arrows point to the first and last lines of the list.