Creating Testbench Files

1. Create your testbench file:
2. Set hierarchy in Quartus Prime
3. Compile in Quartus Prime
4. Open Model Sim
    1. Change Directory to Source Directory of Project
    2. Run the `vlib` command to create a new `work` library, which is the default library where compiled files are stored.
        - `vlib work`
    3. Use the `vlog` command to compile your Verilog files. Ensure you are in the directory containing the Verilog files or provide the full path.
        - `vlog spi_master.v spi_master_tb.v`
    4. Use the `vsim` command to simulate the testbench module. The `-voptargs=+acc` flag can be added for better visibility of internal signals.
        - `vsim work.spi_master_tb`
    5. At the ModelSim prompt, you can add signals to the waveform viewer:
        - `add wave *`
    6. Once the `vsim` command runs, you will be in the simulation prompt. Run the simulation for a specific amount of time:
        - `run 1000ms`
    7. Use the `view wave` command to open the waveform viewer if it isn't already open:
        - `view wave`

Below is a testbench for an SPI implementation in Verilog.

```
`timescale 1ms/1ms

module spi_master_tb();

    parameter bits = 32;

    reg clk_system;
    reg start_transfer;
    reg [bits-1:0] data_inR;
    wire [bits-1:0] data_outR;
    reg [$clog2(bits):0] size_transfer;
    wire cs;
    reg reset_system;
    wire clk_spi;
    wire miso;
    wire mosi;

    spi
    #(
```

```
        .reg_width(bits)
    ) spi
    (
        .clk_system(clk_system),
        .start_transfer(start_transfer),
        .data_inR(data_inR),
        .data_outR(data_outR),
        .size_transfer(size_transfer),
        .cs(cs),
        .reset_system(reset_system),
        .clk_spi(clk_spi),
        .miso(miso),
        .mosi(mosi)
    );

    assign miso = mosi;
    always
        #2 clk_system = !clk_system;

    initial
    begin
        clk_system = 0;
        start_transfer = 0;
        data_inR = 0;
        reset_system = 0;
        size_transfer = bits;
        #4;
        reset_system = 1;
    end

    initial
    begin
        $dumpfile("simple_spi.lxt");
            $dumpvars(0,spi);
    end

    integer i;
    task transact_test;
        input [bits-1:0] data;
        begin
            data_inR = data[bits-1:0];
            #3 start_transfer = 1;
            #4 start_transfer = 0;
            for( i=0; i < bits; i = i + 1)
            begin
                #4;
            end
            #16;
        end
    endtask

    initial
    begin
        #10;
        transact_test( {1'b0, 64'hDEADBEEF} );
        $finish;
```

```
    end

endmodule
```