Using the DE1-SoC

Top-level wrapper for FPGA

In ECE342, you will be able to complete all of your labs using only ModelSim and any text editor. However, some of you may wish to run your code on a DE1-SoC board, if you had previously purchased it for ECE241 or ECE243. This document will show you how to get your code working on the FPGA. For this we will use the `FPGA_kit`, provided to you on Quercus. Please download a copy of this kit for each part of each lab that you want to run on the FPGA and make the changes needed. Reminder to not place multiple Quartus projects in the same folder and to only have a single instance of Quartus running at one time.

To run your code on the FPGA, your code must be instantiated inside a top level module that has the board inputs and outputs (i.e ., switches, keys, LEDs and HEX displays). Please note that this top level module is **NOT** your test bench. Your test bench contains non-synthesizable code (e.g., `initial` block, delay statements such as `#1`) which will not work on an FPGA. The module you instantiate here should be the same one you instantiate inside your test bench.
An example of using the `de1soc_top` module to instantiate a 4-bit adder is shown below. For clarity, code that is different in this example vs. the 'de1soc_top.sv' file you can download is colored red.

```
module de1soc_top
(
    input CLOCK_50,
    output [6:0] HEX0,
    output [6:0] HEX1,
    output [6:0] HEX2,
    output [6:0] HEX3,
    output [6:0] HEX4,
    output [6:0] HEX5,
    input  [3:0] KEY,
    output [9:0] LEDR,
    input  [9:0] SW
);


    // Declare the signals you need to connect the top level signals to your design.
    logic [3:0] input_a, input_b;
    logic [4:0] output_c;
    assign input_a = SW[3:0];
    assign input_b = SW[7:4];

    serial_adder serial_adder_inst
    (
        .a(input_a),
        .b(input_b),
        .c(output_c)
    );

    assign LEDR = {4'b0, output_c};
    // Shows how to display a 5-bit signal, split across two HEX decoders.
    hex_decoder hexdec_0
    (
        .hex_digit(output_c[3:0]),
        .segments(HEX0)
    );

    hex_decoder hexdec_1
    (
        .hex_digit({3'b0,output_c[4]}),
        .segments(HEX1)
    );


    // Turn off the other HEXes
    \textcolor{black}{assign HEX2 = '1;}
    assign HEX3 = '1;
    assign HEX4 = '1;
    assign HEX5 = '1;

endmodule


module serial_adder
(
    input [3:0] a, b,
    output [4:0] c
);
    assign c = a+b;
endmodule


module hex_decoder
(
    input [3:0] hex_digit,
    output logic [6:0] segments
);
    always_comb begin
        case (hex_digit)
            4'h0: segments = 7'b1000000;
            4'h1: segments = 7'b1111001;
            4'h2: segments = 7'b0100100;
            4'h3: segments = 7'b0110000;
            4'h4: segments = 7'b0011001;
            4'h5: segments = 7'b0010010;
            4'h6: segments = 7'b0000010;
            4'h7: segments = 7'b1111000;
            4'h8: segments = 7'b0000000;
            4'h9: segments = 7'b0011000;
            4'hA: segments = 7'b0001000;
            4'hB: segments = 7'b0000011;
            4'hC: segments = 7'b1000110;
            4'hD: segments = 7'b0100001;
            4'hE: segments = 7'b0000110;
            4'hF: segments = 7'b0001110;
            default: segments = 7'h7f;
        endcase
    end
endmodule
```