CS220 Lab#3
Design of Sequential Logic and Simple Arithmetic

Mainak Chaudhuri
Indian Institute of Technology Kanpur

Sketch

- Assignment#1: blinking LED
- Assignment#2: rippling LEDs
- Assignment#3: adding five four-bit numbers

Lab#3

- Make new folders Lab3_1, Lab3_2, Lab3_3 under CS220Labs to do the assignments
- Refer to lab#1 slides for Xilinx ISE instructions
- Finish pending assignments from lab#2 first
- Marks
  - Assignment#1: 2 marks
  - Assignment#2: 3 marks
  - Assignment#3: 5 marks

Assignment#1

- Blinking LED
  - Goal is to make LED0 blink once every one second
  - The on-board clock of FPGA runs at 50 MHz
    - Read Chapter 3 of https://www.cse.iitk.ac.in/users/mainakc/2024Spring/Iec220/ug230.pdf
    - The on-board clock at pin C9 will be the input clock to your design
  - The idea is to send an output of 1 to LED0 once every 50000000 cycles of the clock, keep it on for the next 25000000 cycles, then send an output of 0 to LED0

Assignment#1

- Blinking LED
  - Write a Verilog module that does the following
    - Input is clk and output is led0
    - On every posedge of clk, increment a counter by 1
    - If the counter has reached OFF_TIME, set led0 to 0
    - If the counter has reached ON_TIME, set led0 to 1, reset the counter back to 0
    - Initialize the counter to 0
    - led0 and counter should be declared as reg
    - Use non-blocking assignment for led0 and counter
    - `define OFF_TIME 25000000
    - `define ON_TIME (OFF_TIME*2)

Assignment#1

- Blinking LED
  - Write a Verilog test fixture for ISim simulation
  - Use PlanAhead to assign pins to clk and led0
    - Manually add the following line in the .ucf file

    NET "clk" PERIOD = 20.0ns HIGH 50%;
  - Synthesize the hardware
  - Try changing OFF_TIME and see the effect on blinking rate
  - Submit only the code that has the original OFF_TIME defined

Assignment#2

- Rippling LEDs
  - The goal is to blink LED0, LED1, …, LED7 in that sequence every one second
    - The blinking LED will keep shifting to the left, wrap back, and keep rotating
    - It will look like a ripple of LED light rotating continuously

Assignment#2

- Rippling LEDs
  - Write a Verilog module that has clk as input and led0, led1, …, led7 as outputs
    - All outputs should be reg
    - Initialize led0 to 1 and counter to 0
    - On posedge clk, increment counter by 1
    - When the counter reaches SHIFT_TIME do the following and reset counter to zero

```
led1 <= led0;
led2 <= led1;
…
led7 <= led6;
led0 <= led7;
```

    - `define SHIFT_TIME 50000000

Assignment#2

- Rippling LEDs
  - Write a Verilog test fixture for ISim simulation
  - Use PlanAhead to assign pins to clk and led0, led1, …, led7
    - Manually add the following line in the .ucf file

    NET "clk" PERIOD = 20.0ns HIGH 50%;
  - Synthesize the hardware
  - Try changing SHIFT_TIME and observe its effect on the ripple rate
  - Try initializing multiple LEDs to 1 and see if the whole pattern is rotating
  - Try changing the direction of the ripple to right

Assignment#2

- Rippling LEDs
  - Submit only the code that has the original SHIFT_TIME defined and implements the original shifting pattern of a single LED

Assignment#3

- Adding five four-bit numbers
  - Let the four-bit numbers be A, B, C, D, E
  - We will make use of the four push buttons and the slide switches to take four four-bit inputs
  - We will make use of the rotary push button (ROT_CENTER) and the slide switches to take the fifth four-bit input
  - Read pages 17, 18, 19 from https://www.cse.iitk.ac.in/users/mainakc/2024Spring/lec220/ug230.pdf
    - No need to understand the rotary shaft encoder at this point

Assignment#3

- Verilog modules
  - Compute X=A+B and Y=C+D using two four-bit adders
  - Since X and Y are five-bit outputs (including carry-out), compute Z=X+Y using a five-bit adder
  - Since Z is a six-bit output (including carry-out), compute the final output Z+E using a six-bit adder

Assignment#3

- Verilog modules
  - 1st level: full adder module that can add two bits and a carry-in
  - 2nd level: make use of the 1st level module to design three modules: four-bit adder, five-bit adder, six-bit adder
  - 3rd level: connect two four-bit adders, one five-bit adder, and one six-bit adder to design a module that can add the five inputs
  - The output of the six-bit adder is the final output; show the sum in LED0 to LED5 (LSB to MSB) and carry-out in LED6

Assignment#3

- Write a suitable Verilog Test Fixture for ISim
- Use PlanAhead to assign pins
  - Make sure to assign the correct IOSTANDARD, PULLUP/PULLDOWN, SLEW, DRIVE
  - Manually add the following in the .ucf file
    - NET “PB1" CLOCK_DEDICATED_ROUTE = FALSE;
    - Same for PB2, PB3, PB4, PB5
- Synthesize the hardware on the FPGA