CS220A Lab#4
Rotary Shaft Encoder

Mainak Chaudhuri
Indian Institute of Technology Kanpur

Sketch

- Assignment#1: rotary shaft encoder
- Assignment#2: 7-bit adder/subtractor
- Assignment#3: walk in a square grid

Lab#4

- Make new folders Lab4_1, Lab4_2, Lab4_3 under CS220Labs to do the assignments
- Refer to lab#1 slides for Xilinx ISE instructions
- Finish pending assignments from lab#1, lab#2, lab#3 first

Assignment#1

- Rotary shaft encoder
  - The rotary shaft can rotate in both directions and while rotating it generates two input signals ROT_A and ROT_B
  - Read pages 18 and 19 of https://www.xilinx.com/support/documentation/boards_and_kits/ug230.pdf
    - Figure 2.8 shows how ROT_A and ROT_B signals change as the shaft makes right rotation i.e., A becomes HIGH first and then B becomes HIGH
    - If you read Figure 2.8 from right hand side, it shows what happens when the shaft rotates left i.e., B goes HIGH first and then A goes HIGH

Assignment#1

- Rotary shaft encoder
  - One rotation event is referred to as rotating the shaft to left or right by one step
    - Several steps constitute one full 360^{o} rotation
    - Figure 2.8 shows that one step rotation to the right from detent position (stationary shaft position) causes ROT_A to go HIGH first and then ROT_B goes HIGH
    - Similarly, one step rotation to the left from detent position causes ROT_B to go HIGH first and then ROT_A goes HIGH
    - The changing voltages on ROT_A and ROT_B arise from closing/opening of two mechanical switches
      - These switches suffer from "chatter" i.e., they bounce for a while before settling; leads to train of narrow pulses in ROT_A and ROT_B before they stabilize

Assignment#1

- Rotary shaft encoder
  - The goal of the first assignment is to write a Verilog module that can correctly capture every rotation event and the associated direction
    - At a very high level, a rotation event can be captured by observing the changing levels in ROT_A or ROT_B
      - The challenge is to filter out the chatters so that a false rotation event is not detected (i.e., your module detects a rotation when there is actually no rotation)
    - The direction of a rotation can be detected by understanding the relative order of change in ROT_A and ROT_B
      - The challenge here again is to filter out the chatters

Assignment#1

- Rotary shaft encoder
  - On detecting a rotation event, depending on the direction of the rotation, an LED ripple makes one step progress in the direction of the rotation
  - Structure of the Verilog module
    - Inputs: clk, ROT_A, ROT_B
      - Use the clock generated from pin C9 as you did in lab#3
    - Outputs: rotation_event, rotation_direction (both reg)
    - On posedge clk
      - If ROT_A and ROT_B both are 1, set rotation_event to 1
      - If ROT_A and ROT_B both are 0, set rotation_event to 0
      - If ROT_A is 0 and ROT_B is 1, set rotation_direction to 1
      - If ROT_A is 1 and ROT_B is 0, set rotation_direction to 0

Assignment#1

- Rotary shaft encoder
  - Observe from Figure 2.8 that rotation_direction == 1 means left rotation and rotation_direction == 0 means right rotation
  - Observe that a rising edge in rotation_event signifies a step of rotation
  - Observe that sampling rotation_direction at the time of a rising edge in rotation_event can tell us the direction of rotation
  - Write another Verilog module which exploits these observations to drive LED ripples in appropriate directions

Assignment#1

- Rotary shaft encoder
  - Structure of the second Verilog module
    - Inputs: clk, rotation_event, rotation_direction
    - Outputs: led0, led1, led2, led3, led4, led5, led6, led7
      - Similar to rippling LED assignment from lab#3
      - Initialize at least one LED to 1
    - On posedge clk
      - Copy rotation_event to prev_rotation_event (a local reg initialized to 1) i.e., prev_rotation_event <= rotation_event
      - If prev_rotation_event is 0 and rotation_event is 1 (indicates a rising edge in rotation_event)
        - » If rotation_direction is 0, shift LEDs to right i.e., led0 <= led1, led1 <= led2, …, led6 <= led7, led7 <= led0
        - » If rotation_direction is 1, shift LEDs to left i.e., led1 <= led0, led2 <= led1, …, led7 <= led6, led0 <= led7

Assignment#1

- Rotary shaft encoder
  - Write a top-level Verilog module with inputs clk, ROT_A, ROT_B and outputs led0, …, led7
    - Instantiates the previous two Verilog modules and establishes the connection between them
  - Use PlanAhead to assign pins to the inputs and outputs of the top-level module
  - Synthesize the hardware
  - No need to simulate this design in ISim because ROT_A and ROT_B cannot be given exact input behavior as they will receive from the rotary shaft

Assignment#2

- 7-bit adder/subtractor
  - In this assignment, you will design a 7-bit adder/subtractor that takes two 7-bit numbers A and B in two's complement representation and a one-bit operation code (add=0, sub=1) and produces a 7-bit two's complement result and an overflow bit
  - The input A will be taken in two rotation steps of the shaft encoder with the help of the slide switches (lower 4 bits first and the next 3 bits)
    - Place the switches in position and rotate the shaft one step, detect the rotation event and read in the input bits from the switches

Assignment#2

- 7-bit adder/subtractor
  - After input A is accepted, input B will be taken similarly in two rotation steps of the shaft
  - After input B is accepted, the operation code will be taken from slide switch SW0 in one rotation step of the shaft encoder
  - Internally, you have to count the number of rotation steps to figure out which portion of which input you are currently accepting
  - The 7-bit output result will be displayed in LED0 to LED6 and LED7 should glow only if there is an overflow
    - Carry out is not shown

Assignment#2

- 7-bit adder/subtractor
  - Verilog modules
    - A module to do one-bit addition/subtraction
    - Use the module from the last assignment to detect rotation events
    - Change the module from the last assignment that detects rising edge of rotation events; on a rising edge of rotation event, read the inputs; outside the always block, instantiate an array of seven adder/subtractors and compute the overflow
    - Write a top-level module that instantiates the aforementioned last two modules and connects them
  - Use PlanAhead to assign pins and synthesize the top-level module
  - No need to simulate in ISim

Assignment#3

- Walk in a square grid
  - Imagine a 15x15 grid (length of each side is 15)
    - The leftmost bottom corner is (0, 0)
    - The leftmost top corner is (0, 15)
    - The rightmost top corner is (15, 15)
    - The rightmost bottom corner is (15, 0)
  - A worm is sitting at (0, 0) to start with
  - At each move, the worm can take 0, 1, 2, or 3 steps along east, west, north, or south directions
    - If the move causes the worm to hit the boundary, it stops there
    - For example, at (0, 0) if it tries to move toward west or south, it stays at (0, 0); at (13, 0) if it tries to move toward east three steps, it takes only two steps and stops at (15, 0)

Assignment#3

- Walk in a square grid
  - In this assignment, at each move, the inputs are the number of steps (2 bits) and the direction of the move (2 bits)
  - Use the slide switches to take the four bits of inputs
  - A move of the worm is defined by one step rotation of the rotary shaft encoder
    - Place the slide switches in position, rotate the shaft, and read the inputs
  - After accepting the inputs at each move, compute the new coordinates of the worm using a five-bit adder/subtractor
    - Sign bit is always zero because the operands are always positive

Assignment#3

- Walk in a square grid
  - Display the new coordinates in the LEDs
    - Use LED0, LED1, LED2, LED3 for x-coordinate and the remaining four for the y-coordinate
  - Plan the Verilog modules
    - You can reuse a lot from the previous assignment
  - Use PlanAhead for pin assignment and synthesize the hardware
  - No need to simulate in ISim
  - Hint: use the MSB of the five-bit adder/subtractor's sum output to check if the worm has hit the boundary
    - This is not exactly an overflow because any negative result or a positive result more than 15 would mean that the worm has hit the boundary