Anthony Guzman
Blake Russell
ECE 153A
Prof. Forrest Brewer
October 29, 2021

Lab 2A - The Rotary Encoder

Purpose and expected goals of the lab

Set up a finite state machine that uses information provided by its peripherals, the rotary encoder, to change states and generate events. Since the quadrature encoder is infamous for its high debouncing complexity, the main goal of this lab was to create a better debounced interface for the encoder.

Methodology

Nested switch statements were used to implement the FSM interface. The states of the machine changed based on the input of the pins for the rotary encoder, loosely following figure 1. To account for debouncing pins, a short delay was placed between the interrupt trigger and the reading of the GPIO. For testing, printf statements were used to figure out the states of the encoder, and later LEDs were used for debugging purposes to figure out the changing states of the FSM and the value of the counter. The hardware debugger was also used to check for the waveforms generated by the rotary encoder.

Results

1. Explain how you decode which direction the rotary encoder is twisting in. Also explain how debouncing for the twists of the encoder was done. Show bubble charts (states, inputs and transitions) for the finite state machines in your design.

The state machine in Figure 1 below outlines our design. The FSM used a counting variable to keep track of transitions between states and therefore the direction the rotary encoder was being twisted. The FSM took care of debouncing by only transitioning on a valid count and encoder input. Furthermore, the next state would be differentiated by the current count - this ensures that bouncing behavior of the pins doesn't affect the outcome of the machine.

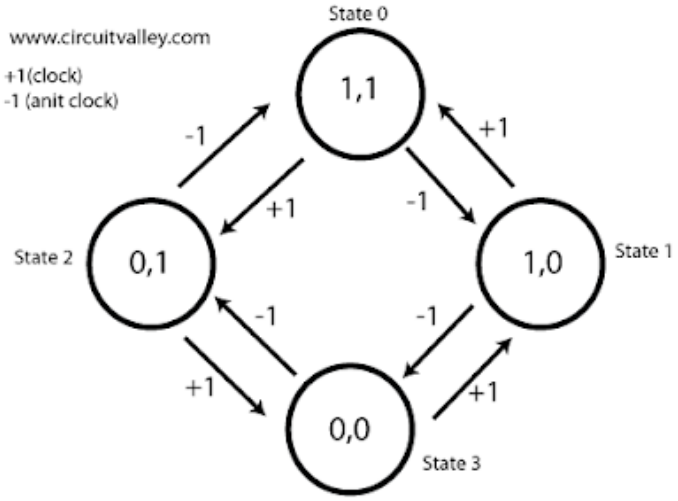

Figure 1. Rotary Encoder State Machine

2. Explain how debouncing for the click of the encoder was done.

The push button requires a timing loop because it will bounce for a short period before settling on a state due to the rebound of the contacts on the physical mechanism. To address the bouncing of the button push, a timer was used to wait roughly 100 microseconds before proceeding. This allowed our system to wait for the bounces and then execute the switch statements of the FSM.

3. Use the hardware debugger to record the waveform of the encoder GPIO change interrupting the processor.

	Delays
Input -> GPIO Interrupts	9 Cycles
GPIO Interrupts -> Processor Interrupt	6 Cycles
Processor Interrupt -> 0x00000010	4 Cycles
0x00000010 -> Interrupt Code	223 Cycles

The results above were obtained from several trials, and averaging them out. The address, 800008dc, was the encoder handler function of the code found from the .elf file. The longest cycle period was from the start of the address 0x00000010 to the interrupt code.

4. What is the Microblaze MSR Register used for? What is the purpose of Bit 30 in the MSR? (Reference: MicroBlaze Processor Reference Guide)

The MSR register is the machine state register. Bit 30 in the MSR is the IE bit which stands for interrupt enable.