Lab 2: Counters, Motor Drives, and Closed-Loop Control Systems

Concepts and Background

Electric motors are a technology that enables much of our world to function. From the brushless DC motor in computer cooling solution to the synchronous motor/generator in a hydroelectric storage dam, motors need some sort of control system to perform its tasks. There are two primary types of control systems: open loop and closed loop. Each receives functions of time as inputs and produces functions of time as outputs. An open loop system receives some input parameter such as the desired motor speed and computes, by means of analog and/or digital circuits, what the input should be to the motor. This is done without regard to the resultant behavior of the motor. "Closing" the control loop requires adding sensors to the motor and/or the mechanical system it is operating to give the controller knowledge about the true state of the system. These measured parameters become additional inputs to the controller that allow it to achieve complex practical results.

Taking the mathematical result from the control system and changing the behavior of the motor requires a device that bridges the low-power computation side to the high-power electromechanical side. The brushed DC motor in the lab kit requires a relatively simple driver. On the expansion board is an STMicroelectronics L6203 integrated full-bridge power transistor motor driver chip. Due to the high inductance of the lab motor, you will control the motor driver using locked anti-phase pulse width modulation. This takes driving scheme takes advantage of the motor's opposition to sudden current changes by applying an alternating-polarity voltage to the motor at high frequency and using the time ratio of the PWM to determine how much the motor is driven in the desired direction. The L620x datasheet illustrates this use of the driver in Figure 8a, referring to it as "Two Phase Chopping."

Implementing a digital logic motor control system will make extensive use of counters. Counters are, in essence, giant state machines with relatively simple inputs and the state assignment matching the numerical output assignment. In addition, the control system will require careful use of Verilog's arithmetic features to produce the desired control function. Since the controller needs to be direction-sensitive, correct use of signed, two's-complement numbers will make completing the implementation easier.

For further information and useful design resources, see:

- http://www.altera.com/support/examples/verilog/ver_behav_counter.html
- http://www.altera.com/support/examples/verilog/ver-add-sub.html
- Quartus II 15.0 Handbook: Example 12-1: Verilog HDL Unsigned Multiplier; page 12-57 Counter HDL Guidelines
- http://www.uccs.edu/~gtumbush/published_papers/Tumbush%20DVCon%2005.pdf

Implementation Requirements

The input stage of this design has two parts: the user input and the feedback input. For the speed goal input, button 0 should count up, button 1 should count down, and button 2 should reset the goal to the zero-motion value. The 'goal' count must not overflow (wrap from max-to-min or min-to-max value) to prevent sudden changes in motor direction. Switch 0 should be used to control the enable input of the motor driver chip, but not change the state of the goal counter or controller. Ensure that the goal count up/down speed is slow enough that a human can set the goal to speeds between minimum and maximum goal. Switches [9:2] should create the 8-bit, unsigned gain constant for use as a parameter of the motor controller. The feedback input is

received from the connector to the motor's quadrature encoder *A* and *B* channels. These square-wave signals will be used to determine the speed and direction of the motor rotation.

To implement velocity detection, sample the relatively slow quadrature encoder outputs (Hz to kHz versus 50MHz FPGA clock) and use the transitions of the signals to drive an up/down counter and periodically sample-and-reset the counter to implement a simple numerical derivative. The reset control signal should be a fixed frequency that is fast enough that the counter cannot overflow when the motor is at full speed, but also slow enough so that the counter accumulates more than ±1 count at low/medium speeds.

The closed-loop DC motor controller will consist of two input variables, one feedback variable, and one output variable. In addition, the controller will take a user-selected constant to change the response of the control system. The control function has the following variables defined:

- g the goal speed
- m the measured speed
- k the gain constant
- r the PWM duty cycle value

These variables will be used to implement the function $r = k * (g - m)$. That is, the output signal is proportional to the current system error. Since this is a digital system, there are practical numerical considerations. Most importantly, the implementation will be limited by the useful bit width of the integers that represent each value. Use division-by-bit-shift to achieve an effective gain range of *0/256* to *255/256* (gain constant of 8 bits, unsigned; bit-shift the multiplication result by 8 bits by dropping the 8 LSBs). The controller should be able to drive the motor in both the positive and negative rotational directions.

The controlled outputs to the motor driver are the locked-antiphase PWM and the enable. As mentioned above, the enable is driven directly by a switch. The PWM frequency must be above the range of human hearing (generally simplified as 20Hz to 20kHz), but must be within the allowable range of inputs to the motor driver.

The computational data path should be 8-12 bits wide, with a constant width throughout. This width is the characteristic of the goal counter output, the speed counter output, and the computational result/PWM input. To avoid certain difficulties in computation, assume that the system error will always have the same sign as the current goal. That is, $(g - m)$ will not have an output overflow/carry bit that is used. This should result in minimum/maximum output duty cycle when the goal is at minimum/maximum if the motor is stopped (disabled or with the rotor held still) and the gain is at maximum.

Demonstration Goals

- Motor speed can be adjusted by changing the speed goal with the three buttons.
- Oscilloscope view of correct PWM square wave shape at safe frequency with correct change in duty cycle if the motor is loaded.
- Steady-state error in the proportional control causes the maximum speed to go down if a lower gain is selected. Note that the design will not have any useful functionality if $k = 0$.
- Disabled motor causes minimum/maximum duty cycle at minimum/maximum goal.
- Speed goal buttons do not cause counter wrap-around.

Code Organization Requirements

- All flip-flops use the 50MHz system clock

- Modularization of design: Verilog modules, one per file, to implement each unit of functionality in your design
- Top-level module contains only wires, buffers/inverters, and module instances
- Assignment of meaningful names to the FPGA externals signals (see the provided motor/encoder signal names as examples)
- Correctly formatted module headers and organized code
- Project submitted on BlackBoard, meeting the submission policy requirements

Design Recommendations/Hints

- Correct behavior is achieved for a gain between 0 and 1, but the effect is only useful for values on the higher end of the scale. We will not engage in further analysis of the control system in this course.
- The motor requires the 12V power adaptor. If the control code changes from one pulse width to another too quickly, the inductance of the motor may trip the protective circuitry inside the adaptor. To reset it, unplug both ends and wait 10 seconds before reconnecting.
- Document the interfaces to your modules. Starting with this lab, you will have code that can be reused in future labs!
- Consider summarizing the specifications so you understand and plan what features you need to implement and to demo.

Design Questions

- What is the control signal frequency specification of the motor driver used on the daughter board?
- What is the chosen output frequency of your PWM generator?
- How frequently does your design determine the motor's rate of rotation (that is, the frequency that the encoder change counter is saved and reset)?
- Assuming a theoretical maximum motor speed of 6000 rpm, what is the maximum motor speed your design will calculate with the provided 192 count/turn encoder attached to the motor?
- Calculate approximately how long it takes to change the speed goal from 0 to full speed in one direction if the button is held constantly.
- At maximum gain, what is the measured pulse width of your motor controller implementation when the goal is minimum and maximum for the free-running, enabled motor?

Report Requirements

- Complete answers to the above design questions.
- Briefly discuss the use of each counter used in your implementation and identify which module it is in.
- Describe the functionality of each module used in the design. Your signal names should make sense relative to their function and correspond to the signal names used in the block diagram.
- Include a block diagram of the connections between the modules and their connections to external signals through the top-level.

- Include oscilloscope captures of both PWM channels at different pulse widths (e.g. stopped, medium/high speed, reverse operation) with brief explanation of the waveforms.
- Summarize how each group member contributed to the completion of this lab.

Grading Distribution

- Demonstration 30%
- Code Organization 10%
- Report Diagrams 10%
- Oscilloscope Captures 10%
- Report Answers and Discussion 40%