Closed Loop DC Motor Control

Nishad Saraf and Chaitanya Deshpande

Portland State University, Oregon.

ECE 544 – Embedded System Design with FPGA

February 27, 2017

Contents

Introduction

The basic idea behind project - 2 is controlling DC motor using PID controller. Out of proportional, integral, and derivative variables for this project for controlling motor, we are only dealing with Proportionality control variable. The peripheral used for interfacing the board to DC motor is Digilent's Pmod HB3 H-bridge driver with feedback inputs.

We have designed custom IP for Pmod HB3. Custom IP has three essential modules one for PWM generation, second for edge detection and the third for handling the communication between Microblaze and custom IP through AXI Bus. Hall effect sensor US5881LUA is used to track the actual speed of motor. At the output pin #3 we will a get square wave which will be feedback to SA pin on PmodHB3. The signed error value viz the actual speed minus the desired speed is then multiplied by KP (Proportional constant). This resultant value is added to the desired speed and then given out to the PWM generation block.

Pmod HB3 Custom IP

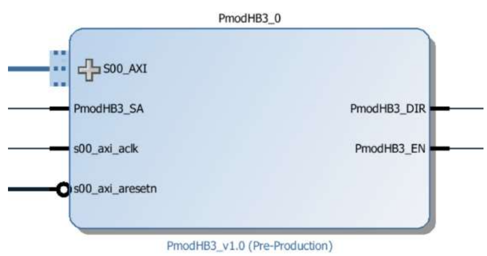

Figure 1: Top view of custom IP

This custom IP has a total for four 32-bit memory registers.

1. Slave register 0: Bit 0 is used to set the direction of rotation of motor.
2. Slave register 1: Low 8-bits are to set the duty of PWM generated wave.
3. Slave register 2: The total edge count of incoming feedback signal from SA is stored.
4. Slave register 3: Reserved for future enhancement.

Three hardware modules handle the entire functionality of IP.

1. **PWM Generation**: Input to this module is 8 – bit duty cycle value while at the output a PWM signal of the required duty cycle is obtained. This value is feed to the enable pin of Pmod HB3. Clock running at 400kHz and reset signal are obtained from AXI Interconnect.
2. **Edge Detection**: Input signal from the third pin of hall effect sensor is feed to this module. Clock running at 400kHz and reset signal are obtained from AXI Interconnect. In a sample space of 1 second the number of positive edge of input signal is counted. Output is 32 – bit count value.

3. **Slave AXI interconnect**: Apart from predefined data exchange protocol input/output signals we have defined one input and two output signal. Clock running at 400kHz and reset signal are obtained from AXI Interconnect. Input is the total count from Edge Detection module while output signals are the duty cycle and direction signal. The duty cycle value taken from the user is stored inside slave register 1 is passed to the PWM generation block. Similarly, the direction bit is directly given out on the output pin PmodHB3_DIR.

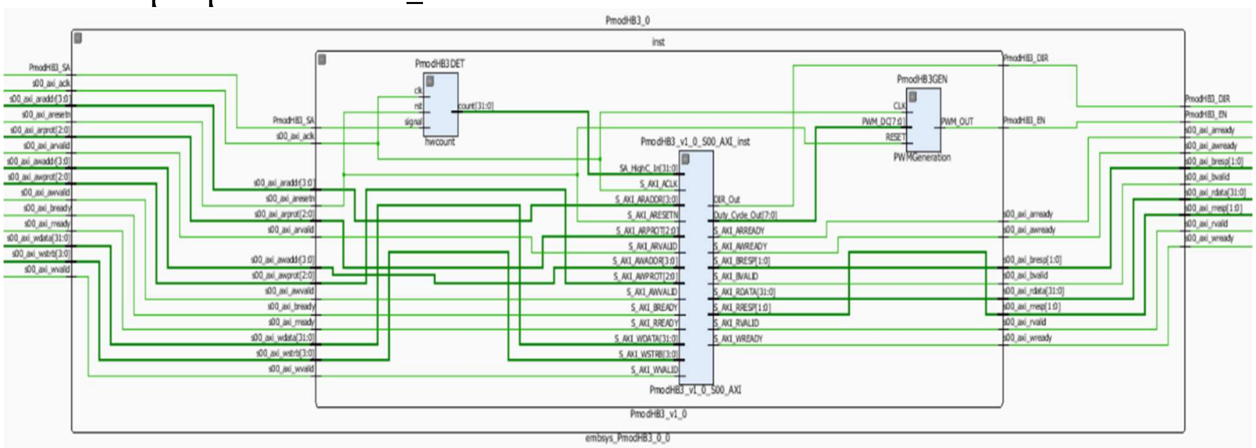

Figure 2: Internal schematics of PmodHB3.

Control Logic

As the count value from edge detection are calculated over a period of one second by directly multiplying this value with 60 gives the total number of rotation in 60 seconds viz nothing but the RPM.

$$Act\quad RPM = 60 \times (Count\ value\ from\ edg\quad detection)$$

The speed that we are encoding through the rotary encoder is a 16 - bit value which is bounded in the range of 0 to 255 in software. This value feed to the PWM generation module as to scale this value to desired RPM it is multiplied by 0.0085 (11500%255…11.5k is the max speed of our DC motor).

```
scaled_rpm = (high_count*0.0085);
actual_speed=(high_count*0.32);
error = ((RotaryCnt-scaled_rpm));
//Error can be negative
dutyRead = dutyRead + (KP*error)+10;
HB3_IP_setDutyCycle(dutyRead);
```

Figure 3: Code snippet of closed loop logic.

Algorithm of Frequency Counter and Feedback logic KP:

1. Start.
2. At negative edge of reset, all counter values are set to 0.
3. Taking sampleCounter for counting positive edges of clock.
4. Another tempCounter for counting positive edges of the signal coming from hall sensor.
5. When sampleCounter reaches to the value equivalent to 1 second then edge counts of the signal stored is tempCounter variable will be updated to the output.
6. By using count value Actual RPM of the motor is calculated.
7. Desired speed of motor is subtracted from actual speed which will then multiplied with KP to get error signal.
8. Then error value is added with desired speed which will be used to set further duty cycle of the PWM signal.
9. End

The green value indicates the value of desired speed, the yellow value denotes the actual speed and the red graph denotes the value of the kP.

Figure 4:Response when speed is decreased from a set value to a smaller value.

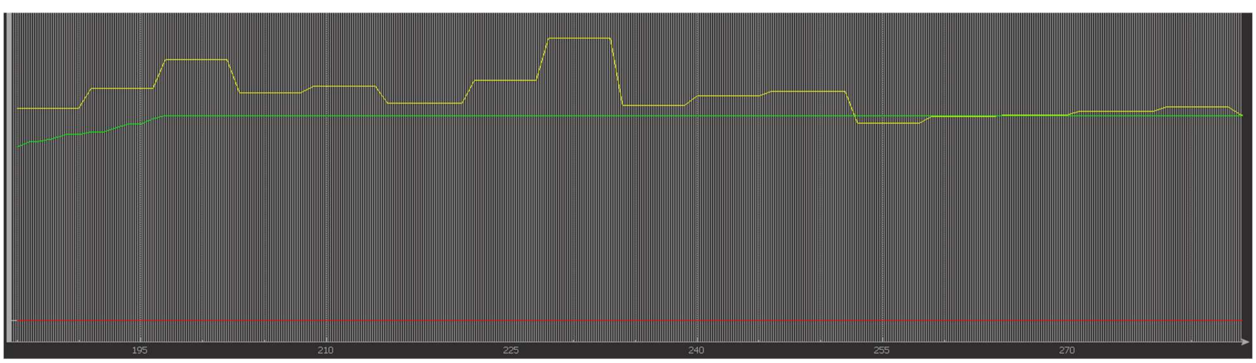

Figure 5:Final steady response.

Flow Chart for Edge Detection

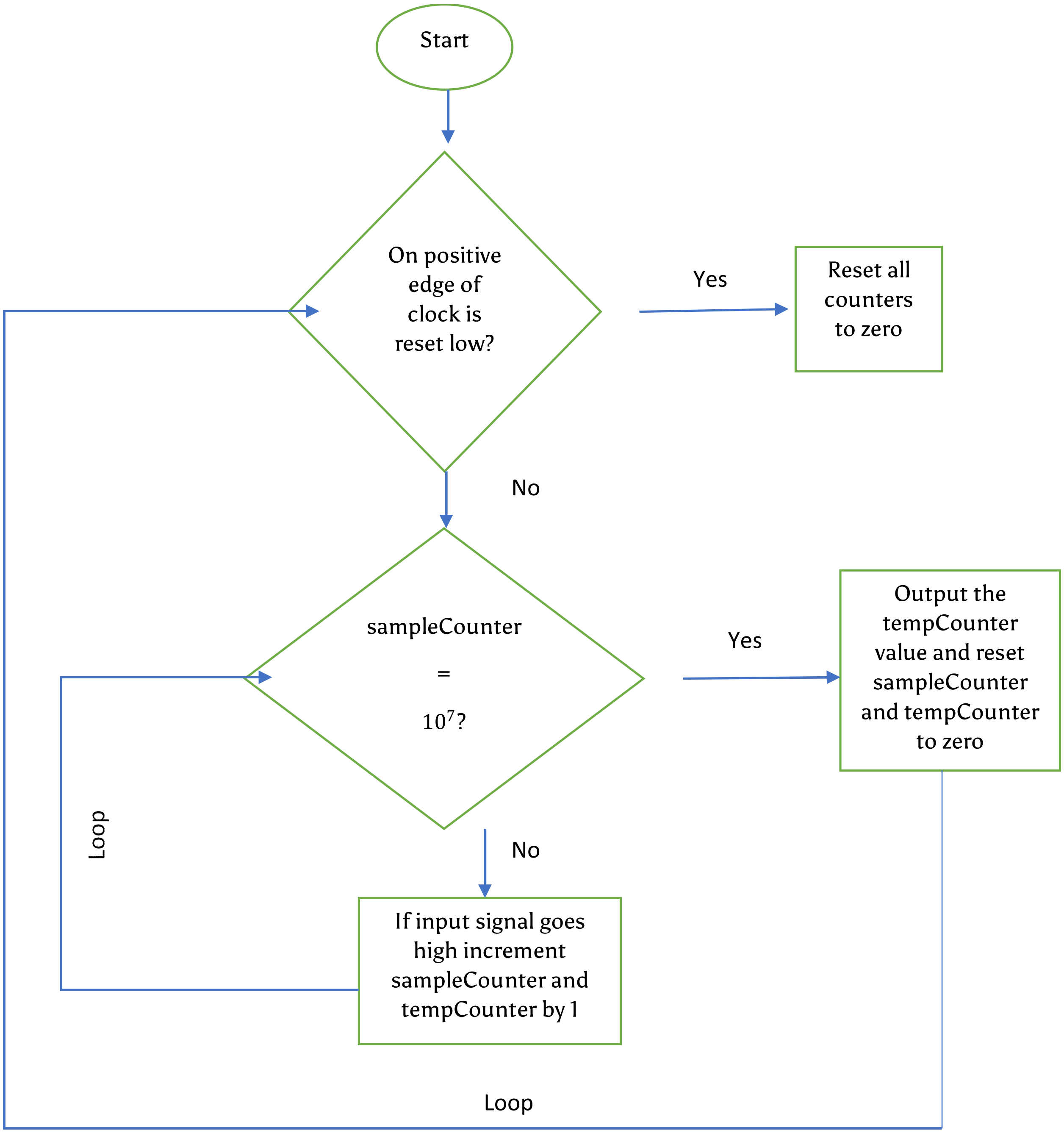

Figure 6: Edge Detection.

Challenges faced

1. While creating custom ip we were confused between input and output ports of the PmodHB3. It took some time to figure out which are the input and output ports of the ip.
2. While updating custom ip we faced a problem such as IP is not found at a particular path.
3. Output of hall sensor contains a lot of noise due to which our count values were fluctuating at the start but after slight changes in algorithm we were able to get stable count values at the output.
4. The system is very sensitive to the position of motor from hall sensors due to which we got incorrect count values at the start but after correct placement of motor we got correct count values.
5. After using two 1000µF capacitors motor is reaching saturation point nearly at 128 value of encoder. So, we remove 1 of the 1000 µF capacitor.

Contribution of team members

- **Nishad Saraf**: Wrote drivers for Pmod HB3, Created custom IP, and documentation.
- **Chaitanya Deshpande**: Wrote edge detection algorithm, worked on Control logic and documentation

Work Borrowed

Aditya Pawar: Pseudo code of display of graph of RPM on OLED.

Conclusion

Close loop feedback control for motor using P control is implemented. For higher values of KP oscillations will occur and motor's desired speed will not match with actual speed even at high frequency. For lower values of KP and at high frequency desired speed matches with actual speed.

References:

1. Lecture notes and project documentation provided of Dr. Hammerstorm.
2. Serial plotter: https://developer.mbed.org/users/borislav/notebook/serial-port-plotter/