Closed Loop DC Motor Control

Nishad Saraf and Chaitanya Deshpande

Portland State University, Oregon.

ECE 544 – Embedded System Design with FPGA

March 6, 2017

Contents

Introduction

The basic idea behind project - 3 is controlling DC motor using PID controller using xilkernal operating system. The peripheral used for interfacing the board to DC motor is Digilent's Pmod HB3 H-bridge driver with feedback inputs. A load motor is connected at the output with resistor and switch. Depending on speed of motor it will drive the load motor. Hall sensor will count the number of rotations of the motor and will give actual speed of motor.

Actual speed of motor is subtracted from desired speed to get an error signal which will be used for controlling duty cycle and hence speed of motor.

Xilkernal operating system is used for project-3 instead of standalone OS. In the project we used four threads first is control thread which will be used for controlling switches and reading rotary count value. Second thread is motor speed thread which is used to calculate speed of the DC motor. Third thread is PID thread in which control logic gets executed i.e. multiplication k_p,k_d,k_i will be done in this thread. Fourth thread will be motor command thread which will be used for setting speed of Dc motor by setting the duty cycle of the motor.

The robustness of the application is achieved by including a watchdog timer that forces a system restart if the application does not periodically restart the timer. The Watchdog timer IP is added to the MicroBlaze design.

Custom IP

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.

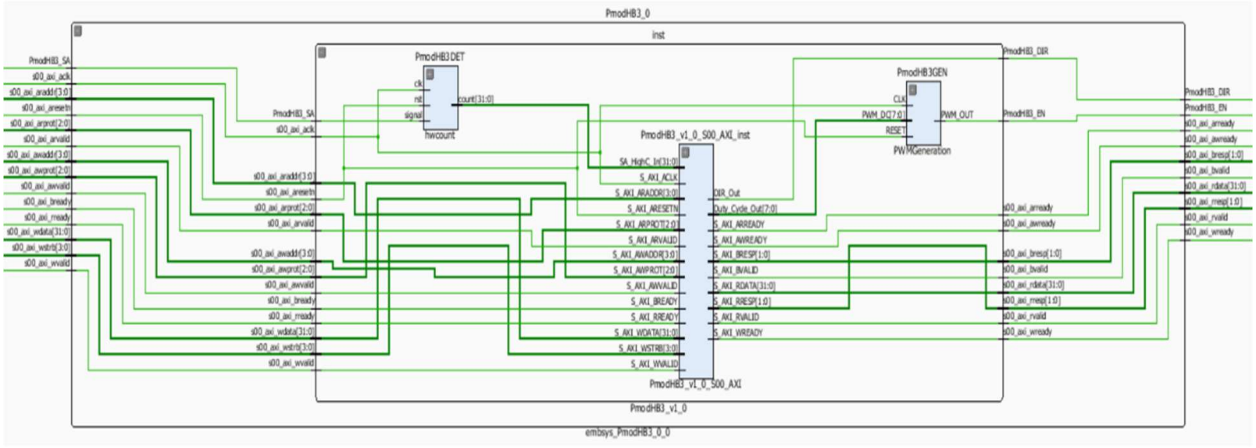


Figure 1: Internal schematics of PmodHB3.

Control Logic

Proportional Control

In case of proportional constant output of the controller is directly proportional to the error signal i.e. difference between actual and desired value of the system. And K_p will be a proportional constant of the P control.

$$\boldsymbol{P_{out} = K_p \times e(t)}$$

P_{out}: Output of P controller
K_p: Proportional Constant
e(t): error signal (desired value – actual value)

Derivative Control

In case of derivative constant output of the feedback system is directly proportional to the derivative of error signal. And K_d will be derivation constant of derivative control.

$$\boldsymbol{D_{out} = K_d \times \left(\frac{de(t)}{dt}\right)}$$

D_{out}: Output of D controller
K_d: Derivative Constant
e(t): error signal (desired value – actual value)

Integral Control

In case of integral constant output of the feedback system is directly proportional to the integration of error signal. And Ki will be integration constant of integral control.

$$I_{\text{out}} = K_{\text{i}} \int_0^t e(\tau)\, d\tau.$$

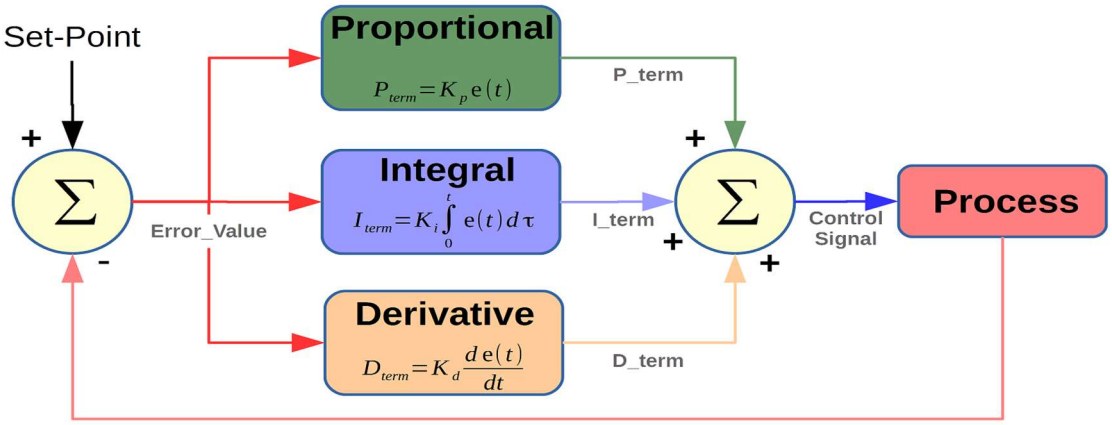


Figure 2: PID Controller (Ref: http://coder-tronics.com/pid-tutorial-c-code-example-pt1)

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.

$$Actual\ RPM = 60 \times (Count\ value\ from\ edge\ detection)$$

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). The PID is controlled individually. The formula calculates the final generated speed: dutyRead + (((KP*error)>>6) + ((KI*iError)>>6) + ((KD*dError)>>6)) +10. This is the final speed obtained after calculating the error.

Code Snippet for Motor control

```
msg_cmd.msg_value = send_dc;   // feedback speed is set
speed = msg_cmd.msg_value;     // feedback speed is used to set motor speed

// Calculation of PID errors
error = (rc - scaled_rpm);
pError = error;
iError = oldError + error;
dError = oldError - error;
incr = (((kp * error) >> 6) + ((ki * iError) >> 6) + ((kd * dError) >> 6));
send_dc = incr + rc + 10;
```

Algorithm of Frequency Counter and Feedback logic PID Controller:

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 to get the error signal.
8. For PID control error signal is multiplied with k_p, k_d and k_i and addition of these 3 takes can calculated.
9. Then addition of multiplication of k_p, k_i, k_d with error is added with desired speed which will be used to set further duty cycle of the PWM signal.
10. 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 k_P.

Figure 3:Response when speed is decreased from a set value to a smaller value.

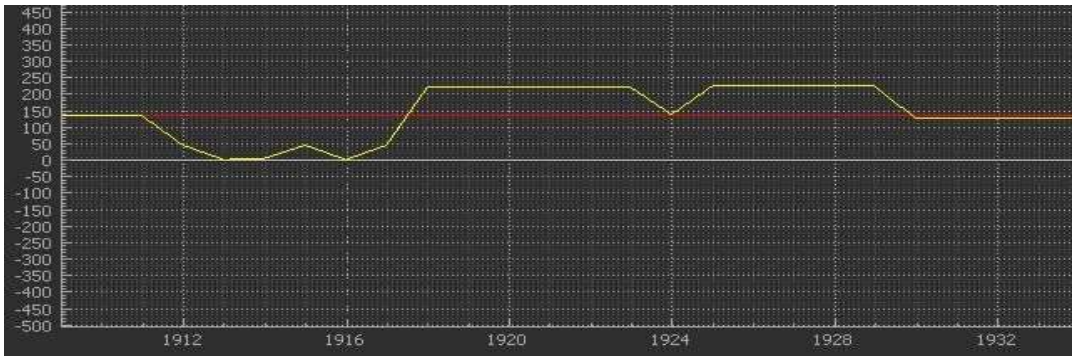


Figure 4: Final steady response.

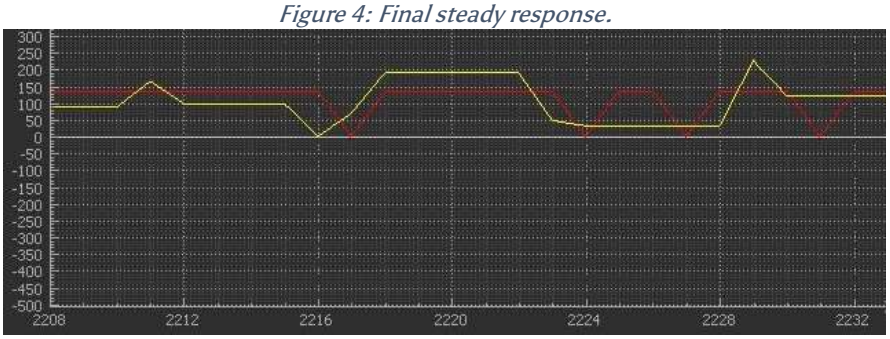


Figure 5: Final PID

Flow Chart for Edge Detection

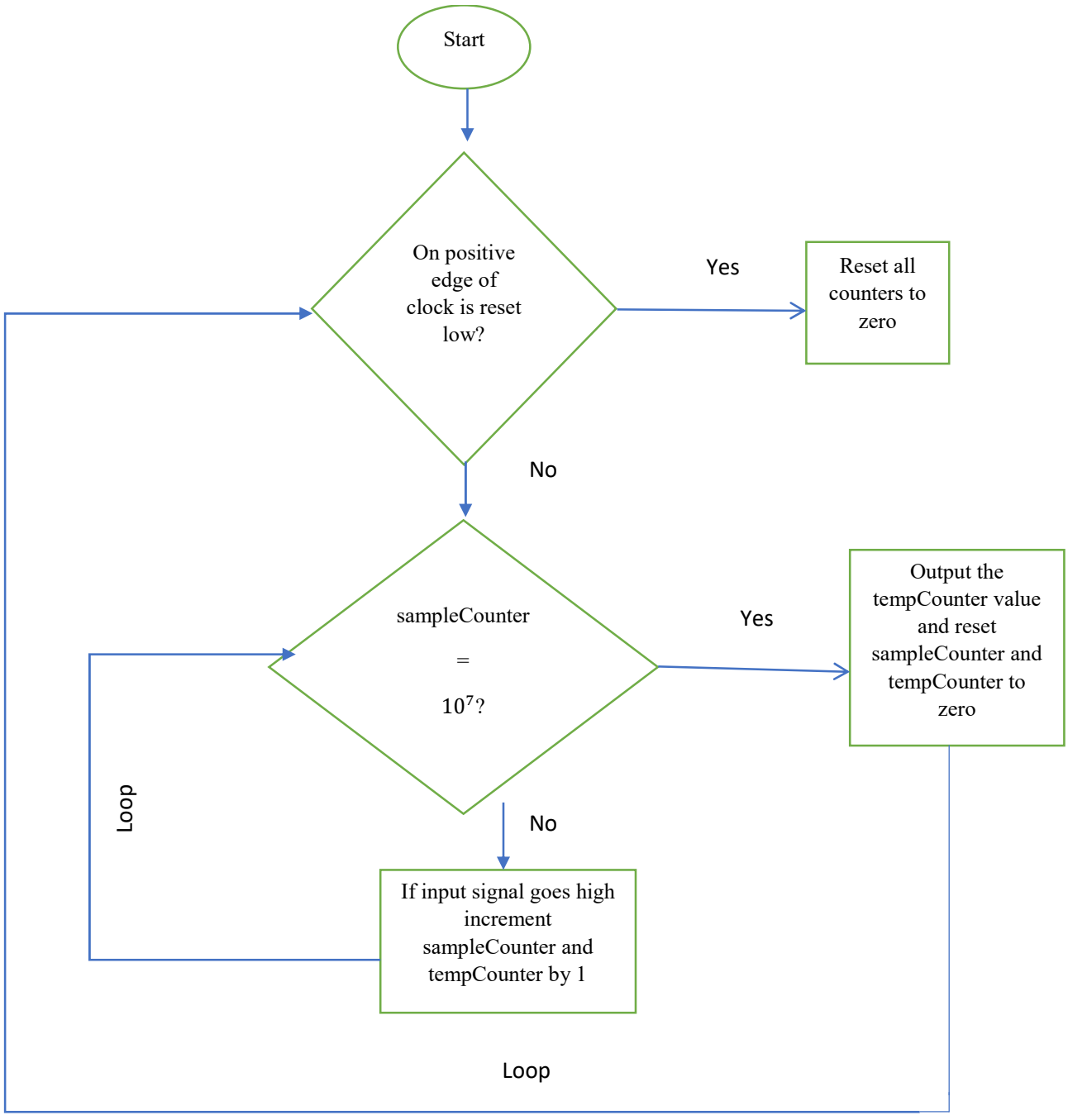
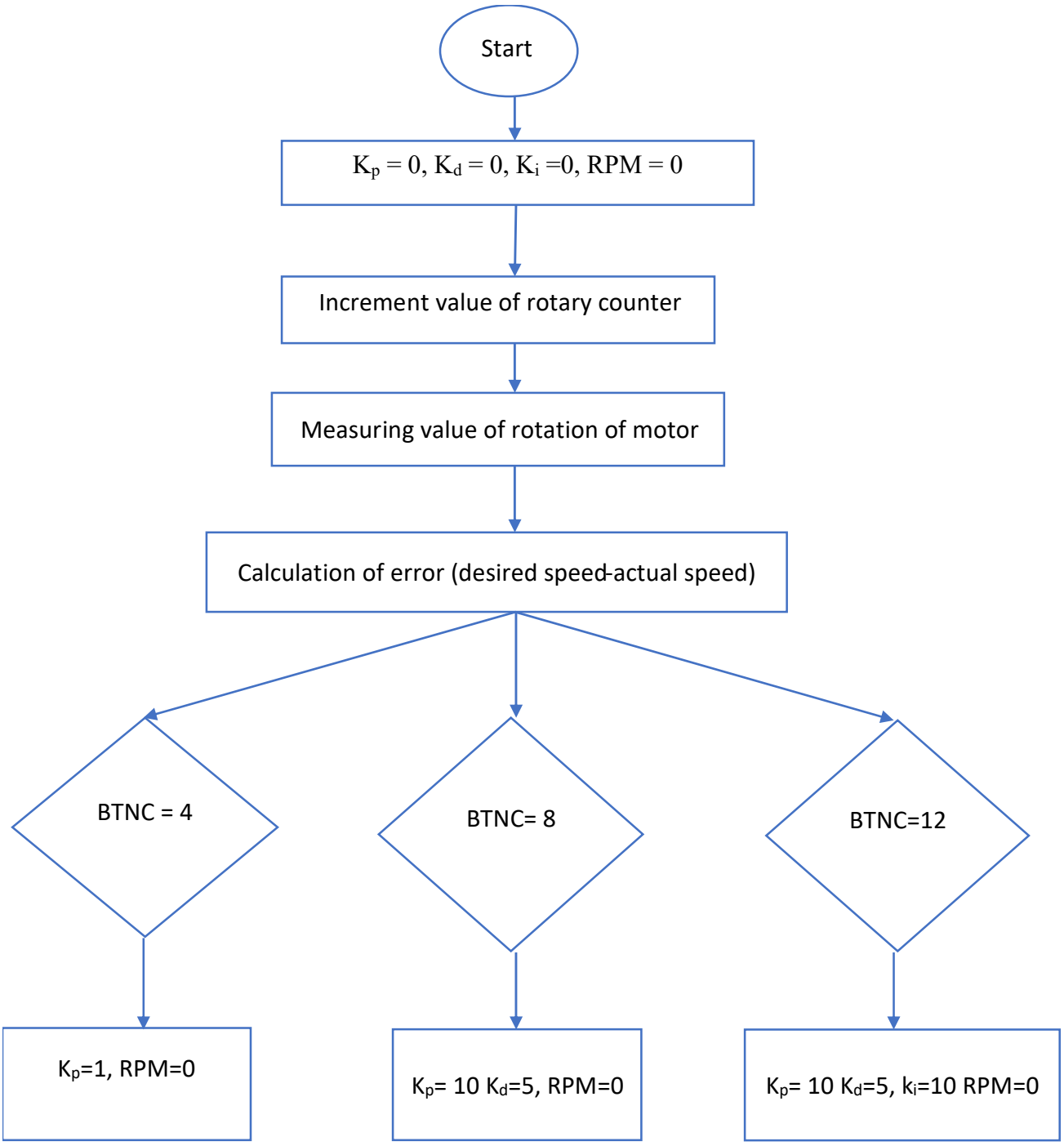


Figure 1: Edge Detection.

Flow Chart for PID Controller



Challenges faced

1. While integrating hardware with the project 3 starter code we faced lots of issue like if we create Xilkernal OS before creating and application project then our application project is unable to switch to that Xilkernal OS.

2. We are tried to print our output speed on PmodOLEDrgb but we faced issue while initializing the PmodOLED module. (For enhancing the functionality)
3. We faced some issue while crashing the watchdog timer but the end we were able to figure out the problem and now we are crashing the watchdog timer.

Contribution of team members

- **Nishad Saraf**: Wrote threads for Motor command and motor Control thread, documentation.
- **Chaitanya Deshpande**: Wrote threads for PID and motor speed thread, documentation.

Conclusion

Close loop feedback control for motor using PID control is implemented using Xilkernal operating system. For values of KP, kI, and KD desired speed will match exactly with actual speed. But after taking Kp, kI and Kd collectively i.e. by using PID control actual speed matches with desired 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/
3. https://www.xilinx.com/ise/embedded/edk91i_docs/xilkernel_v3_00_a.pdf