



Focal Lens Motor Controller Design, Co-Simulation and Implementation on FPGA

Can Uğur OFLAMAZ, MSc.
Electronic Design Engineer
Aselsan, MGEO Division

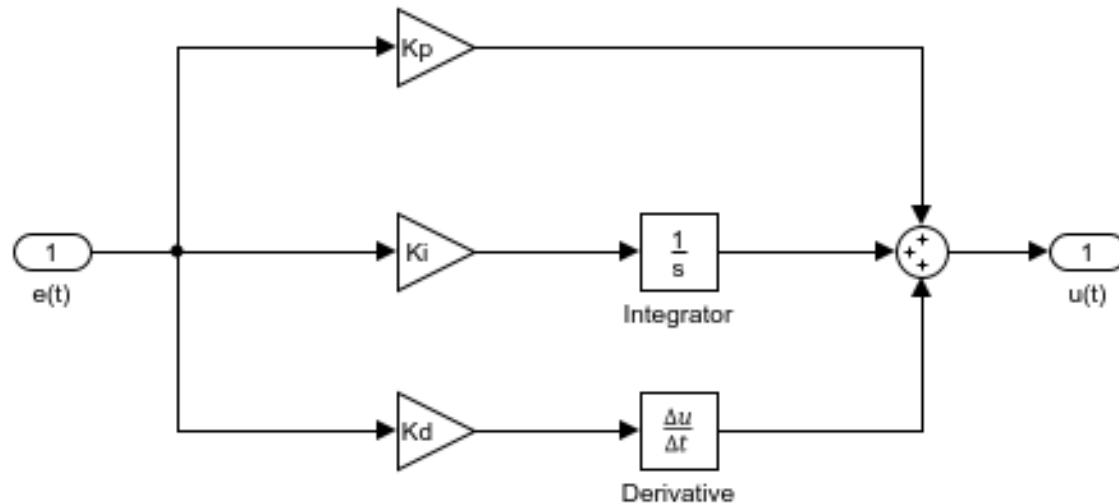
Outline

- ▶ Introduction
 - ▶ PID Controller
 - ▶ Controller Design
 - ▶ HDL Code Generation
 - ▶ Co-Simulation
 - ▶ FPGA Implementation
 - ▶ Verification
 - ▶ Experimental Results
 - ▶ Conclusion
- 

Introduction

- ▶ In this workout a high precision focal lens position controller was designed, co-simulated, implemented and verified for a day vision system.
 - ▶ Encoder coupled brushed direct current (DC) electric motor was used in the system.
 - ▶ The digital controller algorithm was implemented in Field Programmable Gate Array (FPGA).
- 

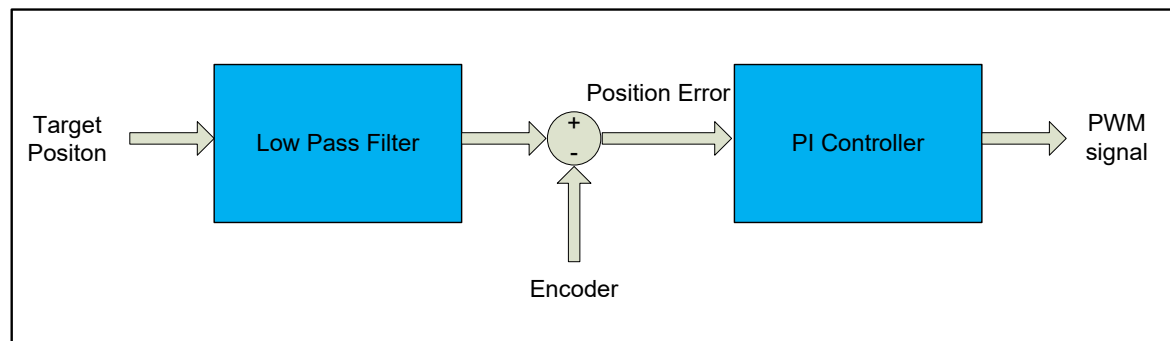
PID Controller



$$u(t) = K_p e(t) + K_i \int e(t) dt + K_d \frac{de(t)}{dt}$$

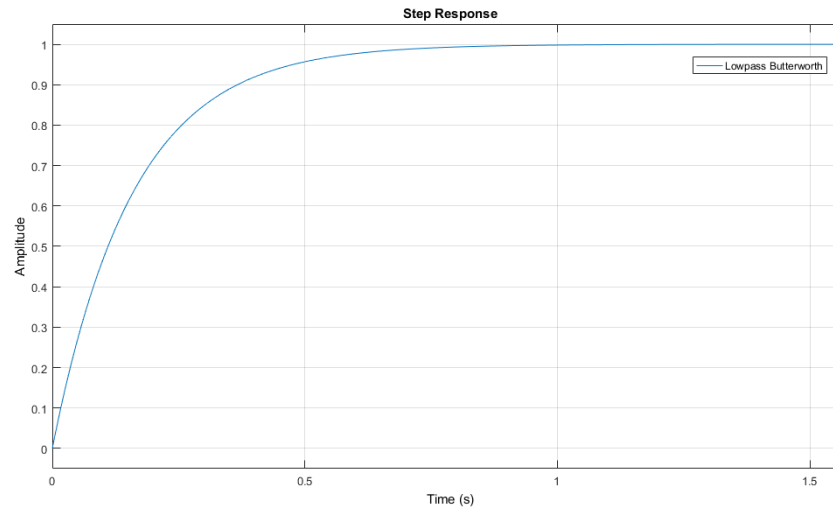
Controller Design

- ▶ The pre-filter and PI controller blocks were designed in MATLAB/Simulink software.
- ▶ Pre-filtering prevents high frequency input commands or soften step commands and reduces the overshoot of control loop.




Controller Design

- ▶ The filter was designed in MATLAB software.
- ▶ First order Butterworth filter structure was preferred for the low pass filter design.
- ▶ The cut-off frequency was set to 1 Hz, and the sampling frequency was selected as 10 kHz.

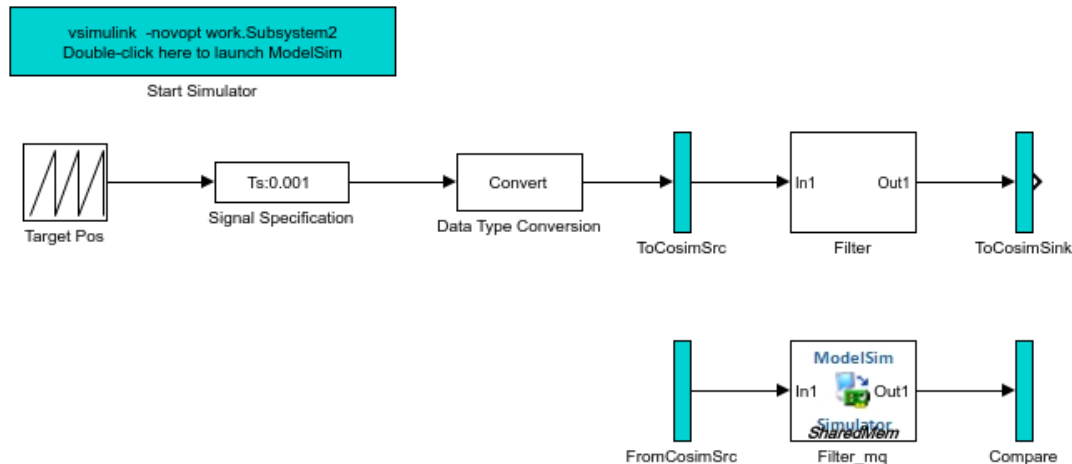


HDL Code Generation

- ▶ By using MATLAB HDL coder toolbox, VHDL codes were generated for the designed pre-filter and PI controller.
 - ▶ Fixed-point arithmetic is used for the code generation process.
- 

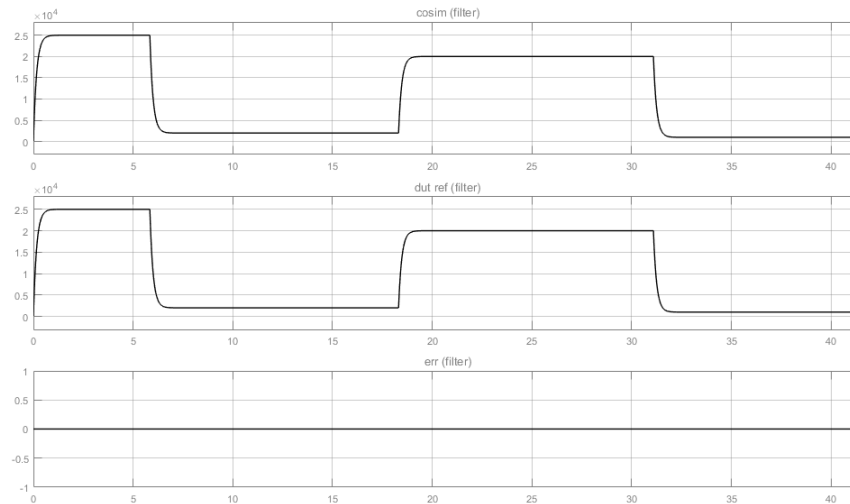
Co-Simulation

- ▶ MATLAB /Simulink was used co-simulate the pre-filter along with Modelsim which is a gate level simulator.



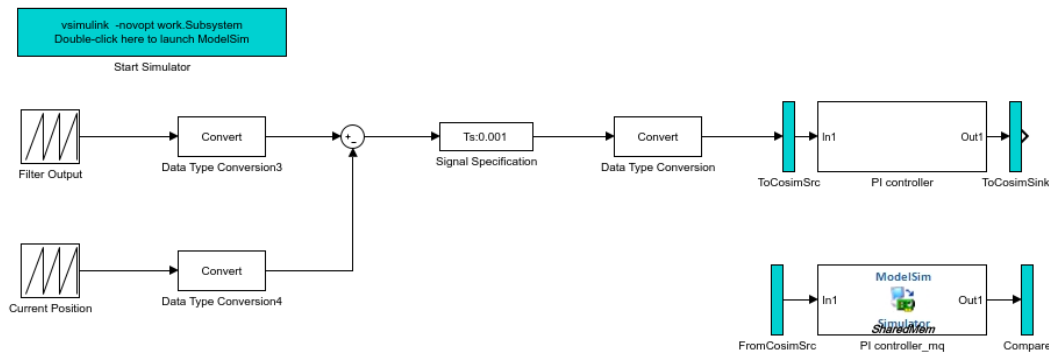
Co-Simulation

- ▶ According to the co-simulation result the output of the Simulink model and the generated VHDL model is perfectly matched.



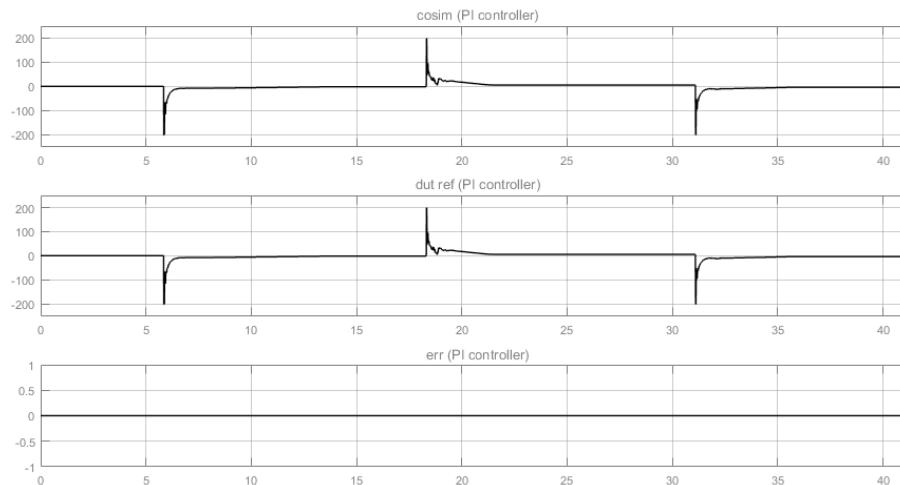
Co-Simulation

- ▶ In the same way co-simulation is done for the PI controller.



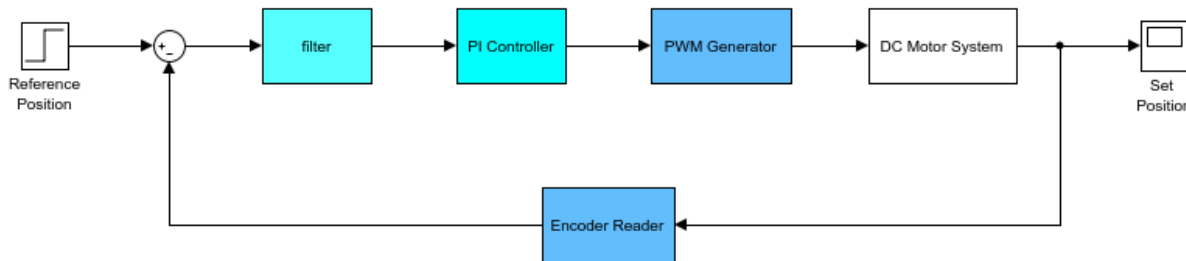
Co-Simulation

- ▶ Again error free co-simulation result is obtained for the PI controller.
- ▶ Eventually, generated VHDL codes can be used for the FPGA implementation



FPGA Implementation

- ▶ Encoder reader and PWM generator codes were written manually in VHDL, but filter and PID controller codes were generated via HDL coder property of the MATLAB.
- ▶ Afterwards, the generated and manually written VHDL codes are synthesized and implemented on the FPGA.



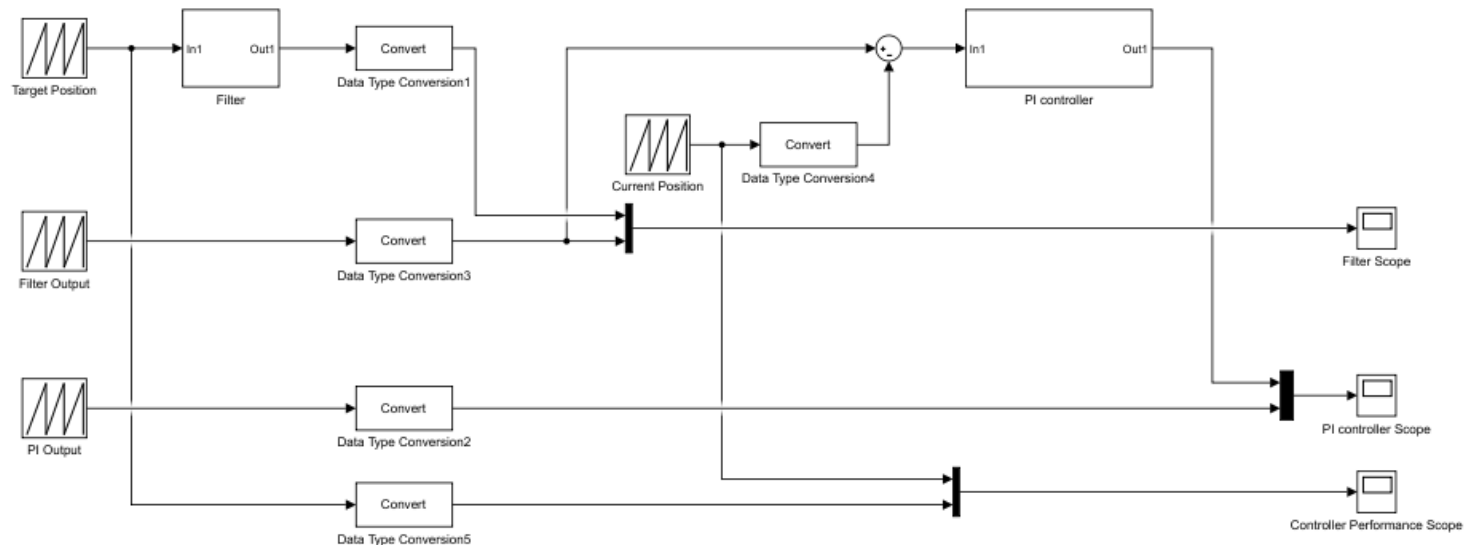
FPGA Implementation

- ▶ The hardware implementation was implemented on a Xilinx Spartan-3AN FPGA.
- ▶ The source usage on the FPGA is shown when the filter and PI controller are used in the table.

Logic Utilization	Used	Available
Number of Slice Flip Flops	167	47,744
Number of 4 input LUTs	411	47,744
Number of BUFGMUXs	2	24
Number of DSP48As	11	126

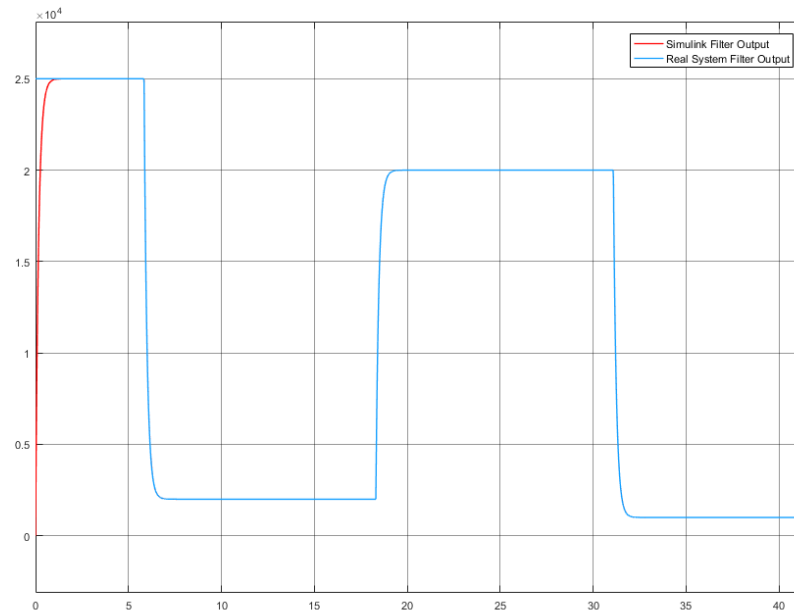
Verification

- ▶ System verification Simulink model is presented in the figure.



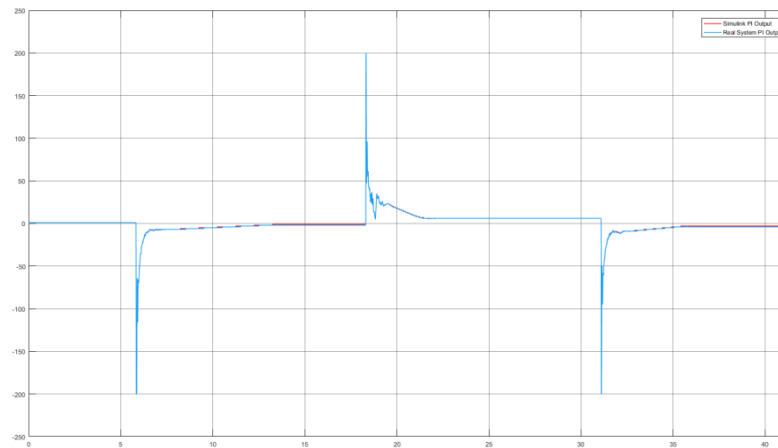
Verification

- ▶ Over here Simulink filter output and real system filter output are compared.
- ▶ Verification result of the filter is shown in the figure.



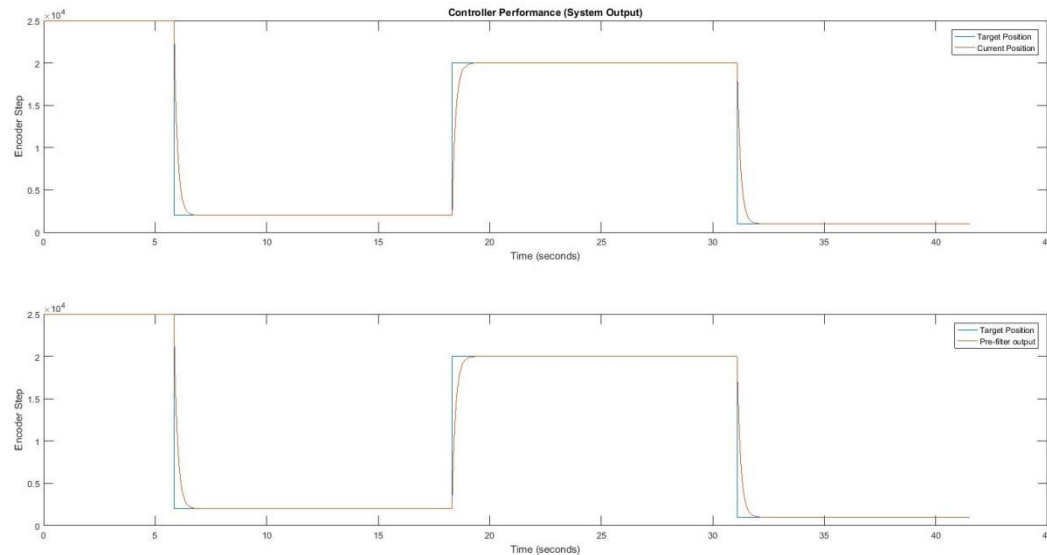
Verification

- ▶ In this section Simulink PI controller output and real system PI controller output are compared.
- ▶ As a result PI controller block is verified as shown in the figure.



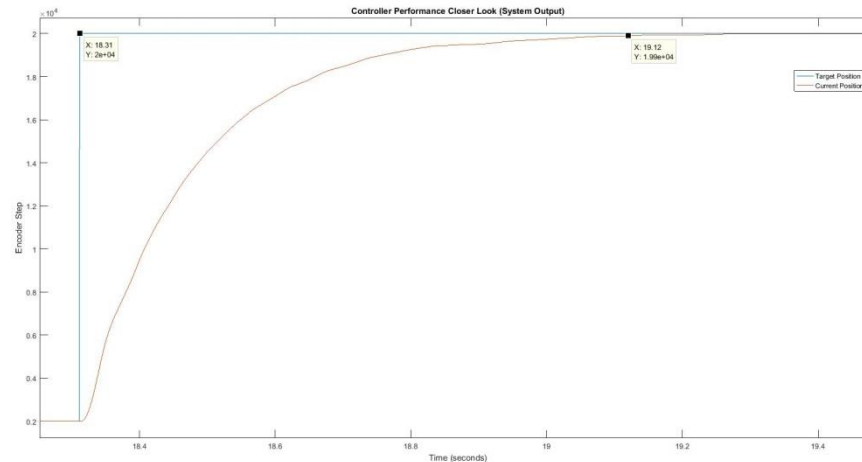
Experimental Results

- ▶ Real system controller performance can be seen in the figure.




Experimental Results


- ▶ Real system controller performance in detail view is shown in the figure.
- ▶ Finally the designed system was tested in $-32\text{ }^{\circ}\text{C}$ and $+55\text{ }^{\circ}\text{C}$ temperature environments.



Conclusion

- ▶ Firstly, the controller was designed which contains a pre-filter and a PI controller.
 - ▶ Then the blocks were converted into VHDL code.
 - ▶ The designed model and generated VHDL code were co-simulated.
 - ▶ Afterwards, the generated and manually written VHDL codes were implemented on an FPGA.
 - ▶ The controller performance was verified by offline simulation using logged real time data.
 - ▶ Finally designed controller system exposed in hot-cold tests.
- 

Conclusion

- ▶ The usage of HDL code generation offers significant advantages such as flexibility, reliability and rapid prototyping.
 - ▶ According to the results and resource usage a quite efficient DC motor position controller system was designed and implemented on FPGA.
- 

Thank you!