PID REGULATOR USING SYSTEM GENERATOR

Ondrej Hock, Jozef Čuntala

University of Zilina, Faculty of Electrical Engineering, Department of Mechatronics and Electronics

Abstract

This article described the implementation of the PID controller in FPGA devices using the Matlab Simulink environment using System Generator for DSP and setting timing parameters.

1 Introduction to System Generator toolbox

System Generator for DSP™ is the industry's leading high-level tool for designing high-performance DSP systems using FPGAs. Highly parallel systems with the industry's most advanced FPGAs are being used. It is possible to provide system modeling and automatic code generation from Simulink® and MATLAB®. It integrates RTL, embedded, IP, MATLAB and hardware components of a DSP system. System Generator for DSP is part of both the DSP and System Editions of ISE® Design Suite. With System Generator for DSP, developers with little FPGA design experience can quickly create production quality FPGA implementations of DSP algorithms in a fraction of traditional RTL development times.

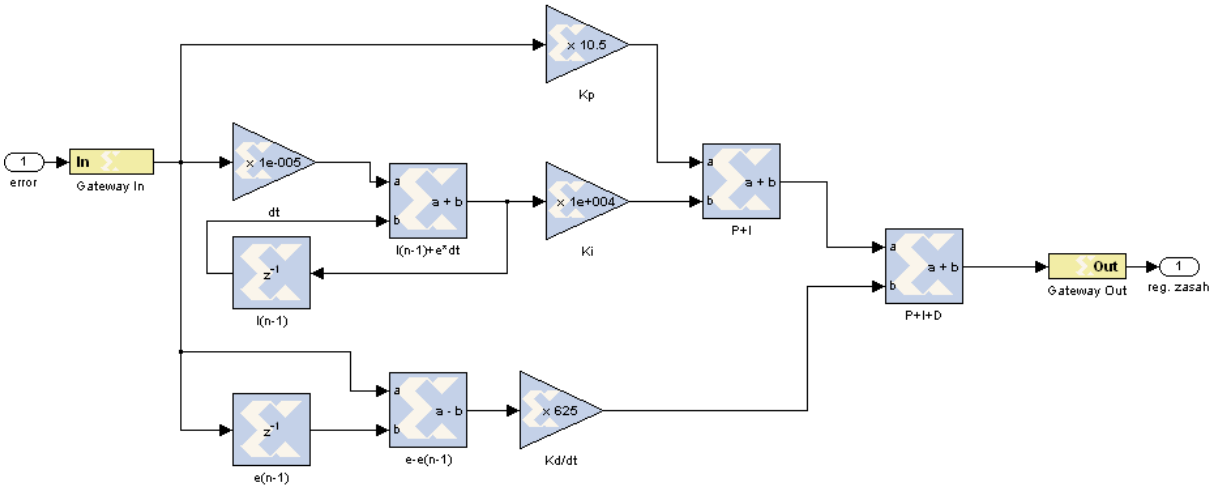

Figure1: PID controller built from System Generator blocks

Figure 1 shows PID controller built in System Generator toolbox. The yellow blocks *IN* and *OUT* creating interconnection between the blocks of Simulink and System Generator blocks. The blue blocks represent the PID controller, which is created according to the equation:

$$reg\_zas = K_P e + K_I \left(I_{n-1} + edt \right) + K_D \frac{e - e_{n-1}}{dt} \tag{1}$$

Figure 2 shows a overall system model in Matlab Simulink. The yellow block is representing the PID controller, which is created in System Generator from figure 1. Block System Generator is important for setting blocks of System Generator.

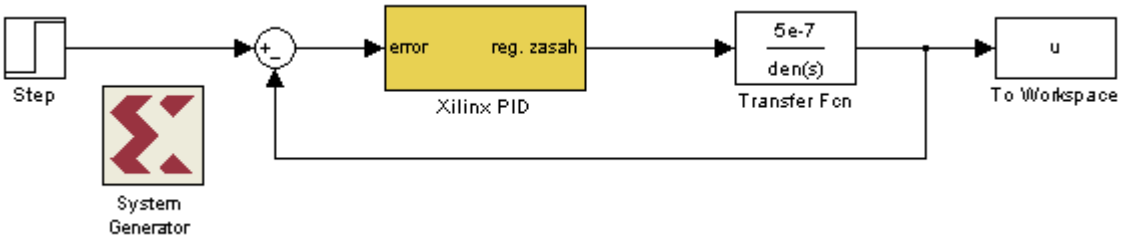

Figure 2: Overall system model in Matlab Simulink environment

Transfer function of system (equation 2) was calculated for the circuit shown in figure 3, from which were subsequently calculated parameters of the controller. Those were after directly used, with a small modification, in the simulation.

$$TransFunction = \frac{1}{LCs^2 + \frac{L}{R}s + 1} \tag{2}$$

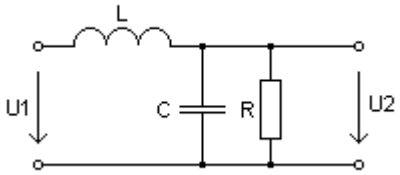

Figure 3: Controlled circuit

2 Understanding of the time constants

There are several types of timing parameters. Those that refer to absolute units such as megahertz or nanoseconds are denoted by upper-case letters. All other parameters are denoted by lowercase letters. Furthermore, the timing measures can be divided into control and analysis parameters. All these parameters are summarized in Table 1.

Table 1: Timing parameters for System generator design

	Name	Symbol	Unit	Effect
Control	Simulation time unit	T_{SIM}	s, ms, ns	Simulation only
	FPGA clock period	T_{CLK}	ns	HW only
	Simulink system period	p_{SYS}	$[T_{CLK}/T_{SIM}]$	Simulation and HW
	Sample period	p_{SAM}	$[T_{SIM}]$	Simulation and HW
Analysis	Sample time	t_{SAM}	$[T_{CLK}]$	None
	Sample frequency	F_{SAM}	MHz	None

The first of the control parameters is the simulation time unit T_{Sim}. It represents our assumption on the fundamental unit of time in Simulink simulation. It affects simulation only. T_{Sim} can be any time unit that suits your needs.

The next parameter is the FPGA clock period T_{CLK}, in the System Generator in units of nanoseconds. It represents the period of the main system clock input to the FPGA from which all other clock and clock enables are derived. Hence, its setting affects only hardware implementation. For instance, for our Spartan®-3A DSP 1800 from Xilinx, the FPGA clock period would be 8 ns (125 MHz).

The Simulink system period p_{sys}, represents the global link between Simulink simulation and hardware implementation. We must set this parameter in the System Generator. During simulation, this value determines how often the System Generator blocks of your model are called, but not necessarily updated, relative to the simulation time unit. For hardware implementation, it specifies the amount of overclocking with respect to the sample rate of the controller. Unlike the System Generator

documentation, we define the Simulink system period as a unit less quantity—that is, the ratio of the FPGA clock period and the assumed simulation time unit:

$$p_{SYS} = \frac{T_{CLK}}{T_{SIM}} \tag{3}$$

This way you can assume any simulation time, as mentioned above. The sample period p_{SAM} in the proposal in SYSGEN, set either explicitly (for example GATEWAY IN-block) or is derived from changes in the speed of the block as raising or lowering the sample. When it is set explicitly, we should enter it as a numeric value in units of the expected time simulation. This setting have affects on the SIMULINK simulation of both the hardware implementation. During the simulation, this value determines how many times the call will block. These must arrive before the block changes state. Therefore, all clock signals in the draft SYSGEN are derived from the FPGA clock input, each period must be an integer multiple of FPGA hour period. However, sometimes FPGA clocks are much smaller than the basic time unit T_{SIM}. Computation time of simulation can be excruciatingly long, given the large number of unnecessary simulation cycles. In that situation you can use different settings for p_{SYS} in a simulation without losing your identity model. This is possible because the value p_{SYS} affects only part of the SYSGEN in your model. Strictly speaking, you can set $p_{SYS} = p_{SAM}$ during simulation of your control system. This ensures that the SYSGEN blocks are called only when needed, and when it actually blocks can change state. Before you generate the FPGA implementation, just set the initial value p_{SYS}.

In the second category of timing parameters, called the analysis parameters, the first is the sampling time t_{SAM}. It serves only for the analysis of the SIMULINK models. T_{SAM} is value determines the hour period in units of FPGAs. The second parameter is sampling frequency FSAM. Each block in SYSGEN displayed F_{SAM} in MHz. FSAM is derived from the other time parameters:

$$\frac{1}{F_{SAM}} = T_{CLKenb} = p_{SAM} T_{SIM} = \frac{p_{SAM}}{p_{SYS}} T_{CLK} = t_{SAM} T_{CLK} \tag{4}$$

where T_{CLKenb} is the period of the associated clock enable in implementation.

From the second equation is obvious that each sampling period p_{SAM} must be an integer multiple of p_{SYS}, because only these clock enable can be derived from the FPGA system clock. Figure 4 exemplifies these relations.

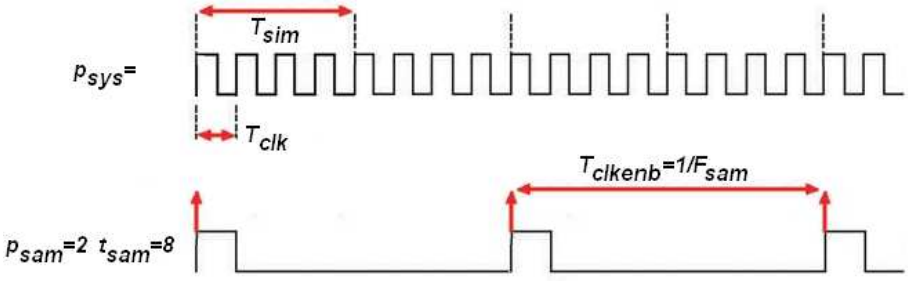

Figure 4: Relation of the six timing parameters exemplified for p_{sys} = ¼

3 Conclusion

After a rather long and for beginners as well as difficult setting of timing parameters, we get the simulation results. Figure 5 shows the response of the regulated system to step response. As we can see overshoot is less than 10%, what is for example the implementation of the PID controller in System Generator is absolutely sufficient. To tune the system according to the requirements, we can just play with parameters of controller.

From the simulation program, we can translate it using SYSGEN to VHDL and used directly in the FPGA. Likewise, we also recovered from the program in a VHDL simulation program. It is also possible to use the simulation program written in M-file, by BlackBox block. System Generator for DSP is a fully-featured tool for simulation programs for the FPGA.

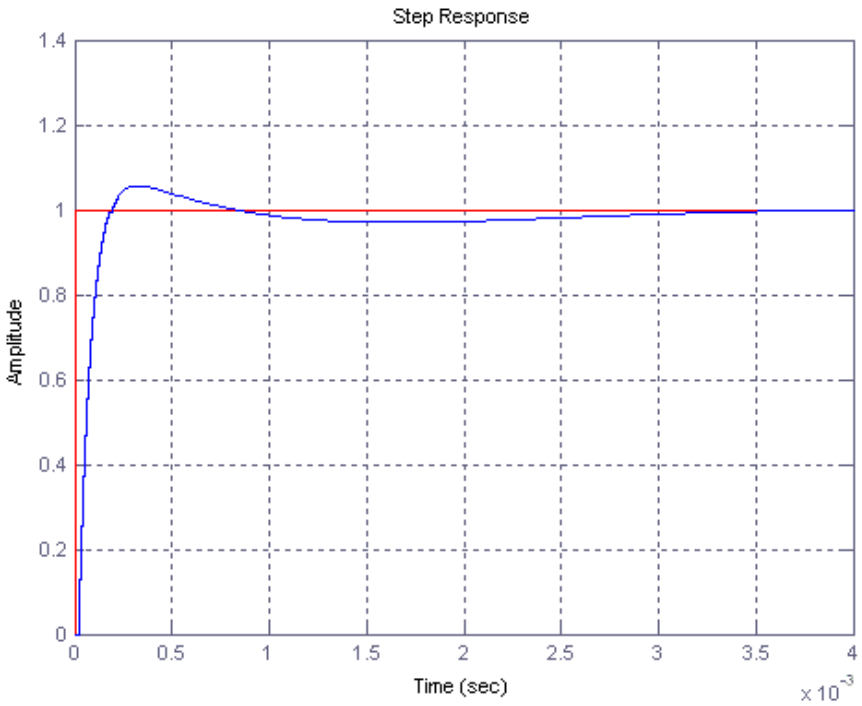

Figure 5: Result of simulation response regulated system to step response

Acknowledgement

The authors wish to thank for the support to the R&D operational program Centre of excellence of power electronics systems and materials for their components, No. OPVaV-2008/2.1/01-SORO, ITMS 26220120003 funded by European Community and also to APVV agency for VMSP-P-0085-09.

References

[1]
[2] *System Generator for DSP performing hardware-in-the-loop with the Spartan™-3E starter kit*, Xilinx, 2006
[3] *System Generator for DSP – Getting started guide*, Xilinx, 2008
[4] www.xilinx.com
[5] K. Khare, R. P. Singh, V. Gupta: *FPGA Design and Implementation Issues of Artificial Neural Network Based PID Controllers*, 2009 International Conference on Advances in Recent Technologies in Communication and Computing
[6] J. Wassner, Ch. Eck: *Understanding timing parameters in Xilinx System Generator*, Xcell Journal, Third Quarter 2009, www.xilinx.com/xcell
[7] L. Qu, Y. Huang, L. Ling: *Design and implementation of intelligent PID controller based on FPGA*, Fourth International Conference on Natural Computation
[8] W. K. Lee, S. Jung: *FPGA Design for Controlling Humanoid Robot Arms by Exoskeleton Motion Capture System*, Proceedings of the 2006 IEEE, International Conference on Robotics and Biomimetics, December 17 - 20, 2006, Kunming, China

---

Ing. Ondrej Hock
University of Zilina, Faculty of Electrical Engineering, Department of mechatronics and electronics, Univerzitna 1, SK-010 26 Zilina, Slovakia, ondrej.hock@fel.uniza.sk,

doc. Ing. Jozef Čuntala, PhD.
University of Zilina, Faculty of Electrical Engineering, Department of mechatronics and electronics, Univerzitna 1, SK-010 26 Zilina, Slovakia, jozef.cuntala@fel.uniza.sk,