

**USB connected Programmable FPGA systems**

**V-II Pro PowerPC®**

- Virtex®-II Pro XC2VP7
- 256 Mbytes DDR Memory
- Configurable digital I/Os
- PowerPC® boot FLASH
- USB 2 or Standalone

**Software Defined Radio**

- Virtex®-II FPGA 1M gates
- 2 ch 125Msps A/D and D/A
- TI C6203 DSP
- 32Mbytes SDRAM
- Configurable Digital I/O
- USB 2 or Standalone

**Imaging with Virtex®-4FX**

- Virtex®-4 FX12 FPGA
- 128Mbytes DDR Memory
- CameraLink connection
- VHDL Imaging Library
- USB 2 or Standalone

Programmable hardware with cables, device drivers, loading tools, examples and Power Supply. Systems can be used connected to a PC using USB, or can function standalone (without USB) using the Initialisation PROMs.

sales@hunteng.co.uk  
+44 (0)1278 760188

**www.hunt-rtg.com**

### Step-by-Step Guide to Choosing Timing Parameters

The example control system developed above offers insights into how to select timing parameters. We have broken the process down into five steps.

#### 1. Identify the plant.

Model the plant by an appropriate transfer function. In our example, we modeled the plant as a PT2 element with a gain factor  $K = 2$ , a time constant  $T = 20$  ms and an attenuation factor  $d = 0.2$ . Hence, the plant is an oscillatory element, as can be seen in Figure 3(a).

#### 2. Choose the simulation time unit.

At this point you can choose the fundamental simulation time unit  $T_{sim}$ , such that the plant transfer function has convenient numerical parameters. In our working example we chose  $T_{sim} = 10$  ms. With the above parameters this yields the plant transfer function

$$H_p(s) = \frac{K}{T^2 \cdot s^2 + 2 \cdot d \cdot T \cdot s + 1} = \frac{2}{4 \cdot s^2 + 0.8 \cdot s + 1}$$

#### 3. Set the Simulink system period.

Given the simulation time unit, we are next going to set the Simulink system period  $p_{sys}$  depending on the FPGA clock period  $T_{CLK}$  of the hardware platform at hand. In the case of the Spartan-3E Starter Kit with 50-MHz system clock, we have  $T_{CLK} = 20$  ns and

$$p_{sys} = \frac{T_{CLK}}{T_{Sim}} = \frac{20 \text{ ns}}{10 \text{ ms}} = 2 \cdot 10^{-5}$$

#### 4. Determine the sample frequency.

A rule of thumb is that the sample rate of the digital controller must be at least 20 times larger than the cutoff frequency of the plant. Our example plant has a cutoff frequency of about 30 Hz and we thus opted for a sample frequency of  $F_{sam} = 1$  kHz.

#### 5. Set the sample period.

Finally, we set the sample period parameter  $p_{sam}$  in the Gateway-In block in front of the controller. In the working example we have

$$p_{sam} = \frac{1}{F_{sam} \cdot T_{Sim}} = \frac{1}{1 \text{ kHz} \cdot 10 \text{ ms}} = 0.1$$

With these settings, we can simulate the model, tune the controller parameters and synthesize the controller logic. However, sometimes the FPGA clock period  $T_{CLK}$  is much smaller than the fundamental time unit  $T_{sim}$ , for instance because your controller is part of a larger design that requires a much higher clock frequency than the controller itself. The simulation run-time can then become unbearably long due to the high number of void clock cycles to be simulated before the controller block actually processes the next data sample. In this situation you can use different settings for  $p_{sys}$  in simulation and implementation without losing the consistency of your model. This is possible because the value of  $p_{sys}$  affects only the System Generator part of your model.

More specifically, you can set  $p_{sys} = p_{sam}$  during simulation of your control system. This ensures that the System Generator blocks are called only when necessary—namely, when the blocks actually can change states. Before you generate the FPGA implementation, you just need to switch back to the original value of  $p_{sys}$ .

Model-based design of closed-loop control systems requires that absolute timing measures of the plant transfer function be consistently related to the timing parameters of your design environment. We have provided a systematic approach to this problem using the Xilinx System Generator for DSP tool.

As a next step, we are going to investigate the hardware co-simulation features of Xilinx System Generator in order to perform real-time analysis of closed-loop systems with FPGA-based controllers. ●