Power Supply Control With FPGAs: Model-Based Design With Matlab, Simulink And DSP Builder

by Peter Markowski, Envelope Power, Chebeague Island, Maine

Digital control has taken the power supply industry by storm and today almost no one needs to be convinced of its merits. But in the low to medium ranges of power and cost, this digital revolution is largely focused on implementations using microcontrollers, DSPs or ASICs, entirely neglecting programmable logic devices like FPGAs (and CPLDs.)

The roots of this situation probably lie in the fact that FPGAs have a reputation for being inherently more difficult to design with and many believe they can be used only by experienced digital designers. After all, not only must they be programmed, they also require setting up the whole logic structure (which with DSPs is fixed in the device) using hardware description languages (HDLs) such as VHDL or Verilog with their strange structures and syntaxes. Because power supply designers make a living by mastering analog circuits (after all, in reality there is no such thing as "digital" power) they have to keep digital complexity at a manageable level. But as this article will illustrate, new devices and design software have advanced, changing the cost/benefit relationship for FPGAs (Fig. 1.)

Model-Based Design is the methodology, which proves to be particularly effective in lowering the barrier to the application of FPGA-based digital controllers in power conversion. Appropriately applied MBD reduces the design time, reduces the number of errors and enables higher performance. (For an excellent primer on Model-Based Design, see the reference.) As a result, the threshold for the feasibility and usefulness of FPGAs is now much lower and programmable logic encroaches on the applications previously reserved for DSPs and ASICs in power conversion applications. This trend is supported by FPGA vendors introducing new models suitable for very low-cost applications.

After spending 12 years designing digital controllers for power supplies using Model-Based Design, I am now convinced that FPGAs are the most powerful and the easiest design platform for many power supply controllers, especially those designs requiring non-standard or high performance.
The lack of a pre-defined hardware structure, which may appear a bit intimidating at first, is in fact the very strength of FPGAs. We can set this structure precisely to the task, as needed, nothing more, without the overhead of mastering the complex and powerful structure of a universal DSP. At the same time, the speed, real-time accuracy, and capability of dealing simultaneously with a very large number of time-critical events is incomparably higher with the FPGA than with the DSP.

In this article I will present an example design of the most-basic power supply FPGA controller. This design requires almost no digital skills, very little time and will follow the principles of Model-Based Design, which can be characterized as follows:

- Multiple, compatible simulation models gradually progressing from the most abstract to hardware specific
- Incremental progress from one model to another to the hardware, so as to break down complexity to easily manageable and "debuggable" steps
- Very close, near-perfect correlation between simulation and expected hardware behavior (greatly enhanced by the merits of the digital technology)
- Gradual progression from simulation to hardware, in particular using hybrid models, operating partially in the target hardware and partially in simulation
- Rapid prototyping—running a closed-loop, real-time controller from a PC while the power conversion is done in real hardware
- Hardware in the loop—running a closed-loop, real-time controller in the target hardware (FPGA or DSP) while the power conversion simulation runs on a PC.

The software used in this article includes MATLAB, Simulink, SimElectronics from MathWorks, and DSP Builder from Altera. The model of a synchronous buck converter with all values is taken from the Sim Electronics demo library. It was chosen because of its availability as well as its simplicity for demo purposes. It can be downloaded in the circuit form similar to Fig. 5.

**Step 1. Fundamental Principle Model With Average Duty Cycle**

According to the classical flow of Model-Based Design we should first build the fundamental principle model (Fig. 2.) This model reflects the fundamental differential equations describing the power plant and controller. It is the fastest to simulate and the simplest in itself but rarely used by engineers and may look a bit odd to many. The diagram does not represent an electrical circuit but rather a set of equations describing physical phenomena.

For example: the output voltage \( V_{out} \) is equal to the output capacitor voltage, which is equal to the capacitor current multiplied by its capacitive impedance \( (1/\text{sc}) \) plus resistive impedance (ESR). The capacitor current is equal to the inductor current minus the resistive load current \( (\text{R}_{\text{load}}) \) minus the step load current (Step Load). The resistive load current is equal to the capacitor voltage times the admittance of \( \text{R}_{\text{load}} \). The inductor current \( (\text{Inductor}) \) is equal to the inductor voltage multiplied by the inductive admittance \( (1/L_{s}) \) plus resistive admittance, etc.

While it makes sense to devote some time to becoming comfortable with this purely mathematical (non-circuit) way of representing power conversion systems, it is not necessary. In the great majority of cases it will be perfectly OK to start with the more-common circuit representation shown in Fig. 5. It is also the form, which can be directly downloaded from the Simulink demo library.

The fundamental principle model is useful for its speed of simulation and for its simplicity. In terms of simulation speed, a 10-ms transient takes only 230 ms to simulate. In this design example, all component values are identical to the Simulink demo (with the exception of added ESR.)

\[
\begin{align*}
L &= 1 \text{ mH} \\
L_r &= 10 \text{ k}\Omega \text{ (parallel)}
\end{align*}
\]
C = 22 µF
ESR = 0.5 Ω
Rload = 3.75 Ω
Istep = 1 A, 200 Hz
Vin = 30 V
Ki = 200 (integral gain)
Kp = 0.5 (proportional gain)

Fig. 2. Fundamental principle model with average duty cycle. Constants shown in model reflect the component values given above.

Applying this model, the output voltage waveform during startup and under 1-A load step looks as expected (Fig. 3.)
An alternative form of the fundamental principle model is shown in Fig. 4. It uses a simpler structure (with fewer feedback loops) and more complex transfer functions. Inductive and resistive parts of the inductor were folded into a single inductor admittance (Inductor) and output capacitance plus ESR plus resistive load were folded into a single composite impedance (Capacitor + fixed load.)

This alternative form of model is presented to show that there are many ways of properly representing the power conversion circuit and Model-Based Design practitioners can chose the one that works best for them. Both models have exactly the same output voltage trajectory as shown in Fig. 3.
Step 2. Technology-Specific Model With Switched-Circuit Simulation

In this next step, the diagram blocks representing mathematical operations are replaced with technology-specific components, in our case electrical. For projects with realistic complexity this would be done in several steps, continuously verifying that the system continues to behave as desired.

Fig. 5 is a representation of the system in which the whole power conversion section was replaced with a typical electrical circuit simulation model while the bottom part, representing the control system, is still implemented in Simulink. Actually, many power supply designers would prefer to make this version their starting point and of course that would be perfectly OK.

Naturally for completeness one can also simulate the compensator using electronic symbols like in Fig. 6, just to make sure that everything is OK. But the actual digital controller will be derived from the s transfer function.
As we compare the simulation results obtained in this step to those obtained in the previous one (Fig. 7), we see there is a small difference in the voltage waveforms. It is due to the transition from the average model (which was used in the fundamental principle model) to the switched one (used in the technology-specific model.) But this discrepancy is clearly within the expected range and confirms the continuity of the design process.

**Step 3. Converting Controller To Time-Discrete Form**

Now, using the Simulink GUI tool we convert the continuous s-domain compensator to a discrete z-domain transfer function (Fig. 8.) There is no need to know the discretization theory, just point and click. For some very aggressive compensation strategies you may consider selecting a particular discretization method, but in typical cases it does not matter at all.
After making this conversion, we run a comparison of the dynamic behavior of the system with the continuous s-domain compensator versus its discrete z-domain equivalent (Fig. 9.). Clearly, the digital controller shows some degradation in the dynamic behavior due to the additional lag associated with the zero-order hold. One can modify the digital compensator directly or go back to the analog version for improvement or start experimenting with a completely different compensator. But in this case, but we will leave it as is.
**Step 4. Conversion Of Controller To Infinite Impulse Response Filter**

Next, a hardware structure realizing the z-domain transfer function has to be found. One of the most popular and natural is the infinite impulse response (IIR) filter. Fig. 10 shows an IIR filter realized for our design example with a simple network of amplifiers, adders and delay blocks with coefficients directly corresponding to the z-transfer function obtained in the previous step.

The methodology of deriving this particular structure of IIR filter directly from the z transfer function goes beyond the scope of this article. (For more on this subject, see multiple references on the Web or contact the author directly.) Alternatively, the "black box" IIR filter from DSP Builder can be used. This ready-made IIR filter is a good choice for simple, linear, fixed compensators.

For high-performance, adaptive, nonlinear, optimal, etc. controllers there is no alternative to a “discrete” compensator, like the one shown. This gives the designer the freedom to modify the coefficients and structure of the compensator during operation, “on the fly,” depending on the conditions and changing regulation objectives. Some examples of such non-standard, high performance controllers will be presented in future articles. Actually, it is precisely this freedom and flexibility of implementing complex control structures, which makes FPGA controllers attractive.

Regardless of the structure used the transient response must be identical to that obtained in Fig. 9.

![Fig. 10. Simulation model with IIR filter corresponding to z-transfer function.](image)
Step 5. Controller Implementation With DSP Builder

Next, the IIR structure is copied using Altera DSP Builder blocks as shown in Fig. 11. Symbols used in this diagram correspond directly to hardware HDL code instantiating the given function. This is the main step, which relieves the power supply designer from the burden of dealing with the nuts and bolts of the HDL.

The generated code is optimized for the specific FPGA fabric (and is not likely to be bested even by an experienced digital designer.) But most of all, it is guaranteed by Altera to be “bit and cycle accurate.” This means that as long as external stimuli are properly modeled in Simulink we will have identical behavior of digital hardware down to every bit and every clock cycle. Later, it can be verified by capturing real-time bit streams in the hardware during real-time operation.

In case any debugging is needed, the typical method would involve having a simulation running two versions of the compensator, one with the Simulink mathematical representation (already debugged) servicing closed loop and another with DSP Builder components in open loop next to it. In this way, stable closed-loop operation can be maintained and representative stimulus generated while the errors can be corrected in “open-loop fashion”
without messing up the main circuit. At the end, when the FPGA block behaves properly (as in Fig. 12), we can swap the two compensators and the whole simulation should have dynamic properties as before.

Absolutely identical behavior of both compensators will occur only if the DSP Builder structure is realized with floating-point precision or at least with a very wide bus in fixed point. That’s because, from the power supply controller’s perspective, even a very small FPGA has an almost unlimited number of gates so that we can very well afford such generosity.

The DSP Builder Advanced Blockset provides multipliers and adders realized in floating-point math. However, for designs with very complex controllers or implementations in a very small CPLD, it may be necessary to use fixed-point precision with the resolution trimmed to the minimum (Fig 13.) It turns out that most of the data path can be trimmed down to around 7 bits before excessive quantization noise and the associated oscillations grow too large.
If the oscillation is too large for a given application, it can be reduced by approximately half with every bit added.

**Step 6. Saw-Tooth Generator And PWM Comparator**

Most basic power supply PWM controllers will be completed with the addition of the saw-tooth generator and comparator as shown in Fig. 14.

In the most straightforward implementation, the saw-tooth generator is based on a free-running counter and thus will be limited to about 5-ns increments in the pulse length (coming from the maximum clock frequency) with the associated limit-cycle oscillations. This should be sufficient for most power-conversion applications. There are also relatively straightforward techniques for increasing temporal resolution beyond the basic clock period, if needed.

**Step 7. Hardware Implementation And Testing**

Once we are satisfied with the simulation of the complete structure we move to the compilation of the DSP Builder file into an FPGA configuration file and programming the actual hardware. It is a straightforward process, which can be performed directly from Simulink/DSP Builder with just a few steps.
What is very important here is that the DSP Builder simulation is "bit and cycle accurate," which means that what we saw in the simulation we should now see on the oscilloscope with great fidelity. If there is a difference it comes from the error in the power processing circuit—either the actual hardware or the previously used simulation model. However, the FPGA digital logic tends to be faithful. I have been using this design flow for 12 years and so far, so good.

Because modeling of the power processing circuitry is fairly accurate and not very complex (at least compared with the model of a nuclear power plant or fighter jet) in most cases it should be possible to go directly from simulation to full hardware implementation including both the FPGA controller and the power circuits.

The rapid prototyping technique may be useful only in the case of uncertainty about the behavior of the plant and the inability to simulate it properly. But even in this case, monitoring the internal buses and registers of the FPGA during actual operation, then reprogramming the device is usually the much better method. Hardware-in-the-loop simulation can be useful if we want to avoid the risk of damaging power processing circuitry due to yet uncorrected errors or because we are testing controller behavior in abnormal situations (especially with large and expensive power converters.)

**Step 8. (An Alternative) Replacing DSP Builder Blocks With HDL Blocks**

In some cases, it may be desirable to convert some portion or a whole digital design into a more traditional and portable form using a hardware description language like Verilog or VHDL. Sometimes parts of the design may already be available in HDL for re-use. In other cases, when an ASIC is the final goal, this conversion will be necessary to go beyond the FPGA prototype.

DSP Builder facilitates easy mixing of blocks of both types without affecting simulation or hardware implementation capabilities. This allows a very convenient and easy-to-debug migration process. Individual modules are modified one at a time and every time simulation of the complete system or/and hardware testing verifies full compatibility of both versions (Fig. 15.)

Yet another approach is possible with Simulink HDL Coder. In this case, a generic Simulink model can be converted to synthesizable Verilog or VHDL directly. This approach in some projects may be appreciated by skillful digital designers but is more appropriate for projects with serious DSP content, not common in power supply controllers.

**Summary**

The design presented here is just an example, and it has been purposefully kept as simple as possible, so as not to obscure the design methodology. It would require very few logic components to implement, occupying only a tiny fraction of an FPGA.
Of course, the purpose of using programmable logic is not to ape the most-primitive analog PWM controller but to enable what is impossible with standard components. Examples of what can be done with digital controllers will be presented in future articles. Ultimately, it will become clear that the flexibility, speed and processing power are so high that the true limitation comes from the power-processing circuits or the engineer’s imagination rather than from the FPGA.

If after reading this article, one gets the impression that the presented method involves a very large number of small steps, that is precisely correct. This is the philosophy of Model-Based Design: break complex control design and implementation tasks into incremental steps with the size and difficulty that can be comfortably and efficiently managed. As accumulated experience and standard libraries grow, the smallest steps get combined into larger ones. But in case something is wrong one can always go back and dig into the details without any discontinuities in the analysis and design process.

Reference


About The Author

Peter Markowski has been involved with power supply design since graduating in 1990 with an advanced degree in power electronics. Most of his career he worked for Emerson, formerly Artesyn and Computer Products as a product designer and advanced technology engineer. Recently, Peter has been involved in high-performance FPGA digital controllers and very high bandwidth, purely switched-mode envelope tracking voltage modulators. This year he started the consulting business Envelope Power LLC. Peter is the author of 11 U.S. patents and several applications encompassing various aspects of the power conversion engineering.

For further reading on digital power control, see the How2Power Design Guide, select the Advanced Search option, go to Search by Design Guide Category and select “Digital Power” in the Popular Topics category.