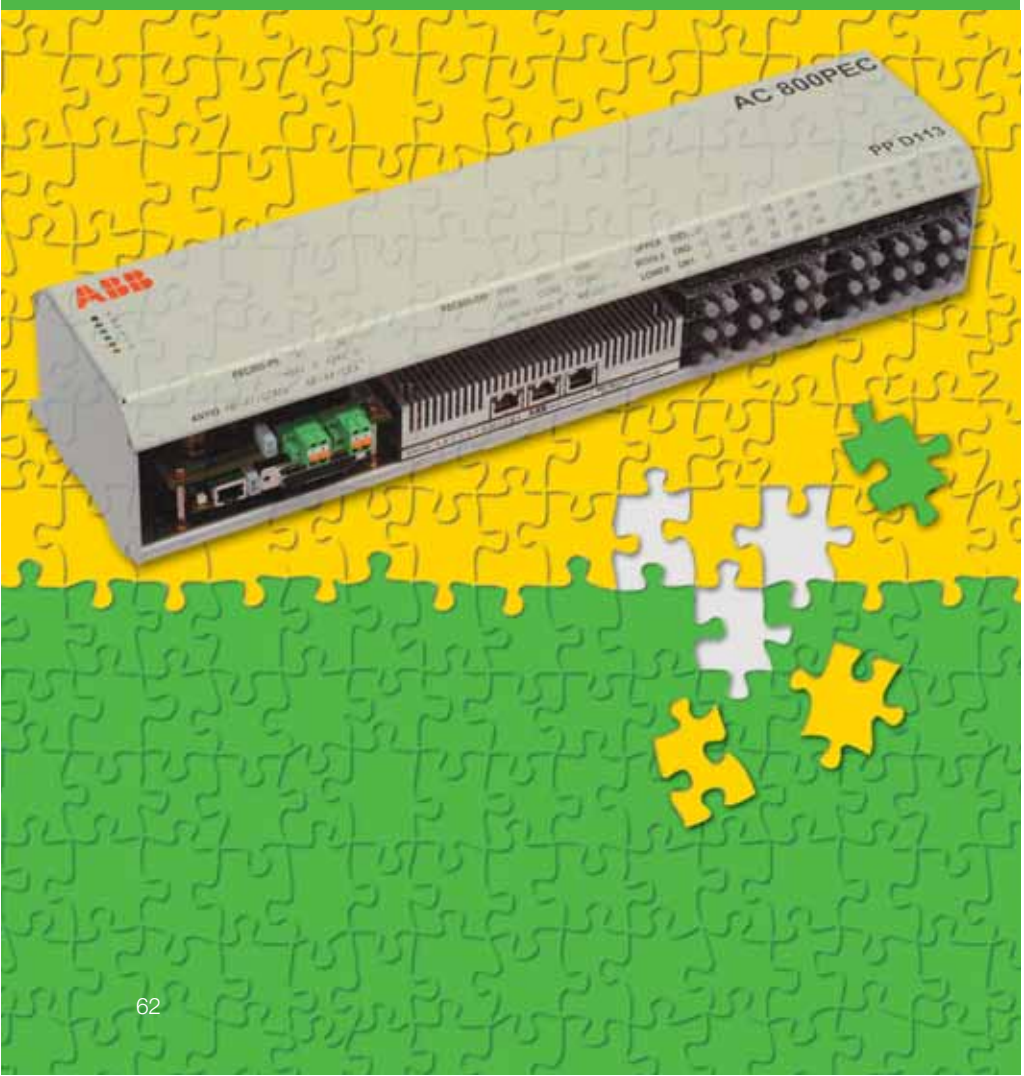
Design patterns

Co-design patterns for advanced control with AC 800PEC

Ernst Johansen

Power electronics has, over recent decades, made great progress, not only in terms of power and speed performance, but also in the breadth of applications being catered for. Power converters are required to become ever faster, cheaper, lighter and more flexible while fitting into less space and requiring less installation and maintenance time.

The implementation of the corresponding power electronics control systems presents many tough challenges, including the magnitude of the control time-domain, which ranges from nanoseconds to seconds. Costs and risks of development can be greatly reduced through the adoption of a control platform. Drawing on tried and tested component technologies, individual systems can be developed very quickly and to high quality and performance standards. ABB's AC 800PEC is such a platform.



Control platforms are necessary to be able to meet the market's demand for faster and more cost-efficient engineering. At the same time, such a platform creates a single-point-of-failure, representing a potential risk to the whole organization. Successful platform development requires striking a delicate balance between optimizing reusability (and so reducing costs) and optimizing performance (at the price of reusability and hence, potentially, at the price of quality).

The secret behind the success of the AC 800PEC control platform is a collection of design patterns that offers excellent testability – a key feature permitting high quality to be combined with reduced time-to-market.

The simulation concept

The concept behind the PEC (Power Electronic Controller) is the development workflow in which simulation models are converted directly into code for the target controller 1. This conversion requires no manual recoding. In this way, an important source of errors is eliminated and a high degree of confidence is provided in the equivalence of the behavior of the simulated and real systems.

The PEC Architecture

In power electronic control, the time-domain ranges from nanoseconds in the switching patterns up to seconds in the start-up sequences. A great strength of the PEC architecture lies in it covering these nine orders of magnitude in the control time-domain without compromising on simplicity or flexibility.

In order to support the direct conversion of simulation models, the architecture 2 has two major differences compared to classic control systems. No dedicated DSP (Digital Signal Processor) is provided for fast control and there is no mechanical rack where I/O modules are connected together. So how does the PEC execute fast control and implement I/O connections?

The control program can be divided into two main tasks: slow control (millisecond range and slower) and fast control. A classic design would utilize two different physical components for these main tasks, a CPU for slow control and a DSP for fast control. By investigating different use-cases it was concluded that the load distribution between fast (typically 100s) and slow (typically 10ms) control was strongly application specific. The lack of a universal rule for load distribution prompted developers to use a single CPU for both fast and slow control. This decision resulted in the need for a very high-performance CPU. Besides solving the load distribution problem, this architecture greatly simplified automatic code generation.

The concept behind the PEC (Power Electronic Controller) is the development workflow in which simulation models are converted directly into code for the target controller.

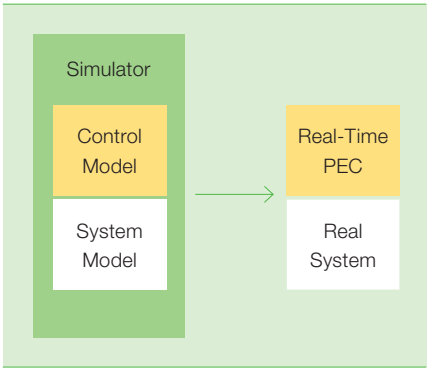
The concept of automatically generating real-time code from simulation models cannot be implemented if the simulation tool offers no auto-code capability. ABB decided to use Mathworks® Matlab/Simulink™ for the system simulation. This tool offers a powerful Real-Time-Workshop™ (RTW) extension for target code generation.

The architecture is designed to support cost sensitive small systems with local I/O only 2c, as well as very large systems requiring distributed I/O 2d using fiber-optic connections. These two system types demand a totally different design of the I/O circuits in the controller. To offer a solution capable of covering all use-cases, a system-level FPGA (Field Programmable Gate Array) was used. This is a hardware component in which the circuit itself is fully programmable. Such FPGAs are used both in the PEC controller and in the distributed I/O nodes. Besides solving the flexibility problem, the FPGA has the additional advantage of being backed up by a very mature design and simulation workflow.

Like the Matlab/Simulink™-based workflow for controller code development, the FPGA implementation workflow is based on a simulator and a compiler. Even thought compilers are available that will translate some Matlab/Simulink™ models into VHDL code, ABB decided not to use such tools in the PEC workflow. The reason for this is that most of the FPGA components in the PEC library are neither modeled nor verified efficiently in the Matlab/Simulink™ language. Instead, a VHDL-based workflow was used for the digital circuits. The adopted workflow was originally developed for ASIC design where high first-past yield[1)] is mandatory. Furthermore, the workflow offers excellent modeling and verification capabilities.

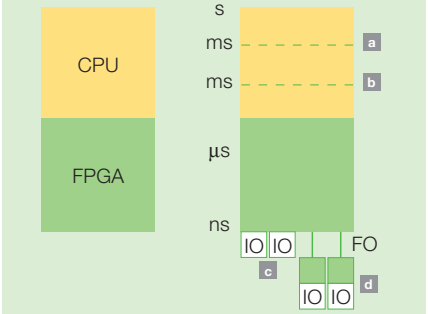
At the time the architecture was defined, however, there was one major drawback – the cost of the high-performance CPU and the system-level

1 The simulated model is automatically converted to executable code for the real-time domain

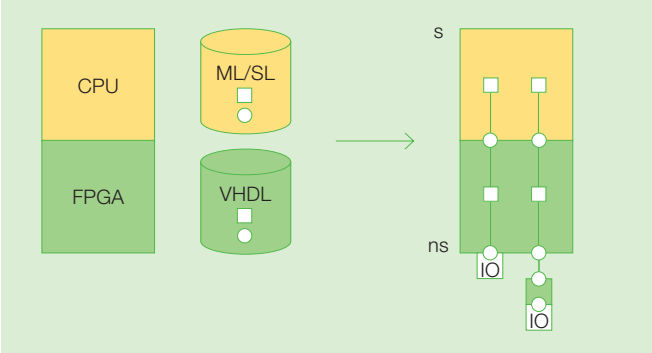


2 A single model can flexibly be adapted to handle different control time-domains

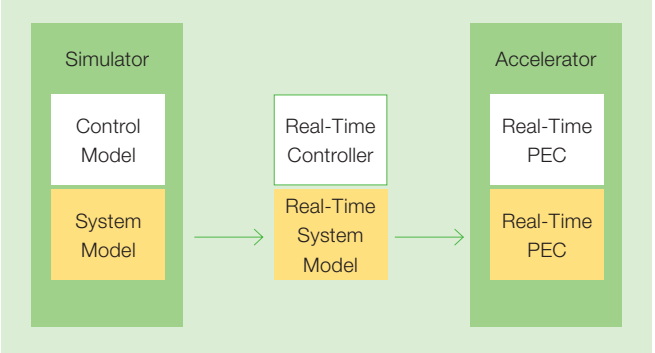
a mainly fast control
b mainly slow control
c Local I/O
d Distributed I/O connected by fiber optics



3 Co-design patterns defined by ML/SL (Matlab/Simulink) and VHDL models



4 System models are converted into the real-time domain for accelerated verification through execution on PEC hardware



FPGA. How this problem was finally solved will be shown later in this article.

Design patterns for control and verification

A design pattern is a pre-engineered solution-template to a specific problem. Design patterns are a method that has been used by software engineers for a long time. However in the area of hardware/software co-design, the definition of generic patterns is more difficult [1]. The AC 800PEC control system makes use of the design pattern method for several design issues found in power electronic applications. A collection of reusable design patterns allows development engineers to rapidly define new systems with high complexity. A system engineer can concentrate on solving his unique problem while trusting in pre-engineered patterns for implementation details.

PEC systems differs from most other systems in that the design patterns in the PEC system are not pure software patterns, but reusable co-design patterns 3. The motivation for using co-design patterns is to cover nine orders of magnitude in the control time-domain (ns to s), a capability not feasible using one single technology (eg, software).

Co-design is, however, a great challenge for system verification. Excellent test coverage is mandatory to assure high confidence that the implementation is error-free, but the simulation of a control system covering nine orders of magnitude in the time-domain is extremely slow. Simulating a complete PEC co-design system would take days and weeks to complete on a PC workstation. Such a prerequisite is simply not compatible to fast time-to-market requirements.

But the PEC concept has an intrinsic feature that can be used to solve this tricky problem very elegantly: The concept behind the PEC is to offer a workflow where simulation models are converted directly into target controller code. This principle is not only applicable to the control model, but also to the model of the simulation environment used with it. By executing the control and system model on the PEC controller concurrently 4, the verification of co-design patterns in the real-time domain is speeded up significantly.

Co-design – a real challenge for embedded system designers

A signal filter can be implemented using analog electronic circuits, a digital filter in an FPGA, or as a piece of software running on a CPU. These solutions all offer identical functionality, but differ totally in terms of cost and reusability. Co-design is about taking the right decisions on how to map a solution to different technologies.

The invention of system-level FPGA components meant that programmability was no longer restricted to software. The invention permits new design patterns for hardware and system design. As there is no cookbook for co-design, it remains a real challenge for the system designer.

Excellent test coverage is mandatory to assure high confidence that the implementation is error-free

System simulation to explore optimal design patterns

In the process of finding optimal algorithms and structures, system simulation is applied in the evaluation and comparison of different designs. As an example of the co-design process, the Analog-Digital Conversion (ADC) circuit is discussed in the following.

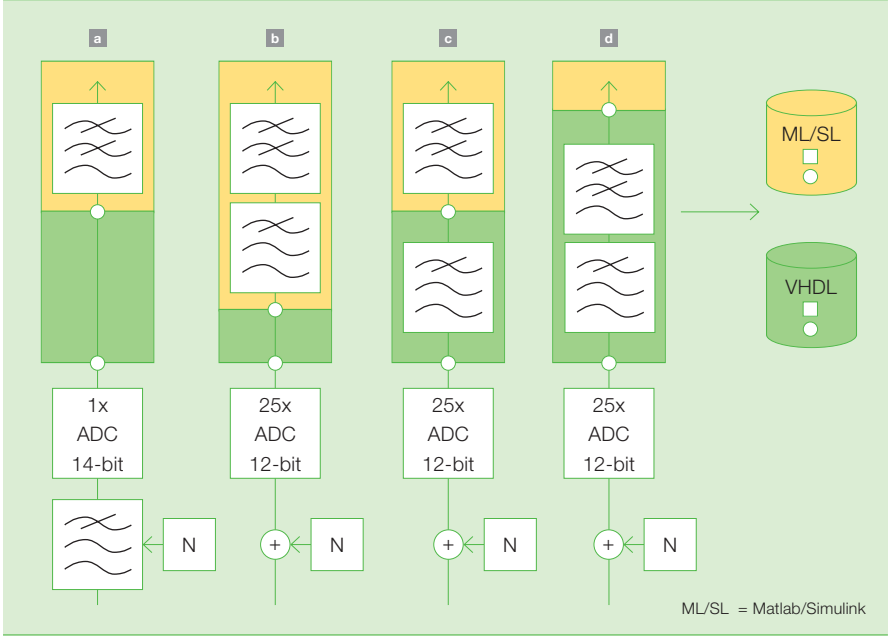
As the developers were required to improve the existing ADC design pattern in terms of cost and quality (Signal to Noise Ratio – SNR), they selected different topologies 5 that fitted the PEC architecture. The topologies where simulated in the Matlab/SimulinkTM simulation environment and compared in terms of complexity and quality.

The developers concluded that, theoretically, the best SNR was obtained by utilizing a combination of oversampling and digital filters 5a (due to the noise-shaping capability of digital filters [2]). Over-sampling 5b-d utilized a much-lower cost ADC circuit than this solution, but added the need for a high-speed digital filter operating at 25x-speed. Was it feasible to implement the filter? Should the filter calculations be executed on the CPU or in the FPGA? Did it pay-off to increase the digital processing payload?

Direct Code Generation

The capability to automatically convert simulation models into real-time control applications made it very easy to create target code for the different

5 Analog-digital conversion co-design topologies, with different components of the task being handled by analog circuits, on FPGA and on CPU



topologies. As the PEC had a build-in load monitor it was easy to measure the CPU load (payload) for all topologies 6. Operating the fast filter 6b in software turned out to generate a too high CPU load and was not feasible.

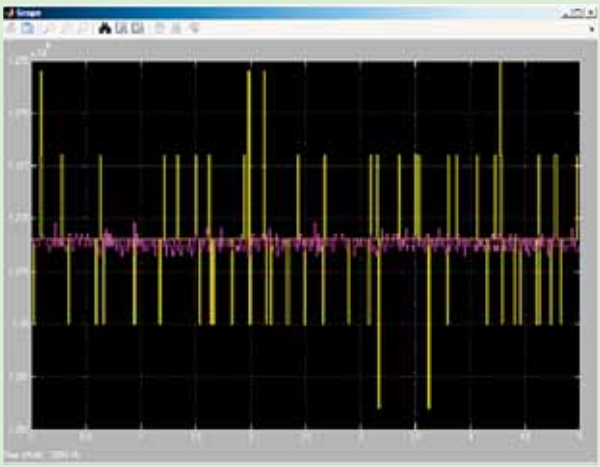
As Matlab/Simulink™ offers comprehensive libraries it was actually not necessary to develop any new code for the CPU filter design.

Optimized VHDL Components

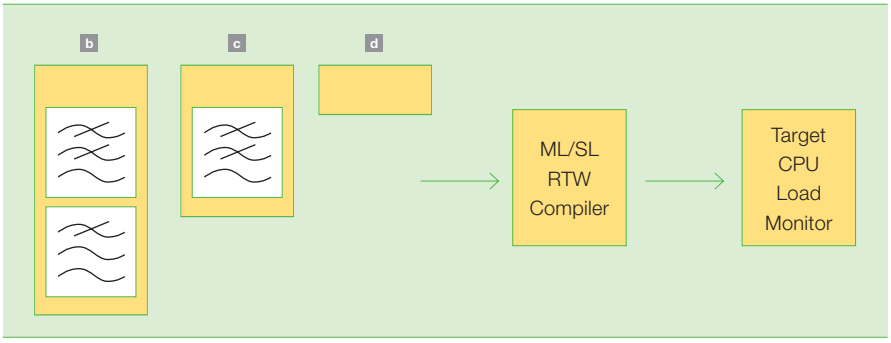
For the FPGA filters, Matlab/Simulink™ was used to evaluate the filter topology, characteristic and calculate the appropriate coefficients. The implementation and simulation of the filters was done in the VHDL environment.

In an FPGA circuit, the payload is measured in circuit area. Compared to a digital filter implemented on the CPU, FPGA filter design offers many more options. The precision (number of bits), the clock-frequency, the filter architecture, the throughput (samples per second), the number of MAC (Multiply-Accumulate) operations per filter and the number of channels per filter are all programmable, offering a vast choice of design alternatives, all with different payloads. The 5c topology, with one high-speed filter operating inside the FPGA and one slower filter calculated by the CPU, turned out to offer the most cost-efficient co-design solution. This was selected as the preferred design pattern for ADC conversion 7.

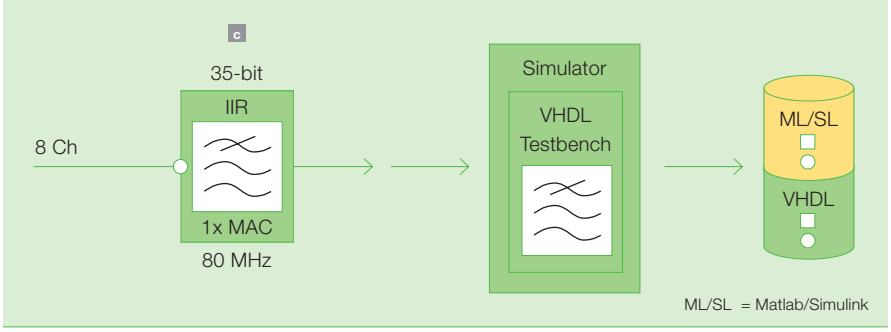
8 Real-time verification of 12-bit / 1MSps ADC (yellow) and FPGA-filter with noise-shaping (pink)



6 Target load evaluation of variants 5b-d



7 Optimal VHDL filter pattern (variant 5c)



Real world verification

During the co-design process, the real system was modeled – including expected signal noise. In many systems, the noise is unpredictable and the simulation of noise unreliable. Real world verification is therefore still important to guarantee product quality 8.

Cost and performance – a moving target

At the time of the definition of PEC architecture in 1999, the drawback of the architecture was the high cost of the CPU and the system-level FPGA. As these components where very expensive at that time, they where used primarily in high-end applications such as flight-simulators and prototyping systems for ASIC development.

As the process technology for digital circuits improved very rapidly, the manufacturing costs of CPU and FPGA dropped dramatically – during a period of five years the cost of these digital circuits was reduced by more than 90 percent. As these lower-cost devices came onto the market, a further advantage of the architecture paid off – its excellent application portability. Today ABB is offering AC 800PEC controllers based on the most cost efficient 90 nm silicon process technology, offering customers excellent product quality at a very competitive price.

Ernst Johansen
ABB Schweiz AG
Turgi, Switzerland
ernst.johansen@ch.abb.com

References

[1] F. Mayer-Lindenberg, Dedicated Digital Processors: Methods in Hardware/Software Co-Design, John Wiley & Sons (February 12, 2004), ISBN 0-470844-44-2

[2] Walt Kester, Analog-Digital Conversion, Analog Devices Inc. (March 2004), ISBN 0-916550-27–3, 2.37–2.41

Footnote

[1] First pass yield is a ratio of the number of "good" units (ie, not requiring rework) to the total produced.