Some new tools give you the ability to co-simulate the RF, analog, and DSP portions of digital communications systems.

Carter Smith, Hewlett-Packard, EEsof Division

Developing an Environment for Mixed RF and DSP Design

Traditional analog communications systems are slowly and steadily being pushed into oblivion by digital systems. Modern communications systems, however, can't shake their reliance on RF technologies, and they've become an inextricable mix of RF, analog, and DSP components.

Digital signal processing plays an indispensable role in these systems as a way to achieve higher data rates and increased bandwidth efficiency. But just as important, DSP techniques have become essential in overcoming some of the problems that the RF and analog nature of these systems presents.

Communications systems, such as cellular phones, are plagued with problems such as multipath or propagation channel distortion, phase noise, RF standing wave distortion, quantization errors, and amplifier nonlinearities, to name the most significant. DSP designers can't work in ignorance of these real-world RF and analog problems if they want to achieve the multiple goals of minimizing bandwidth, reducing power consumption, and providing high-quality communications.

A Unique Environment

The need to deal concurrently with interacting RF, analog, and DSP technologies places some very tough demands on design automation tools. Unlike designers of pure analog or pure digital systems, wireless designers need tools that let them model and co-simulate analog, RF, and DSP circuits in a single development environment.

In addition, designers of these mixed systems need a methodology and tools that let them exchange design information smoothly among the engineers involved in subsections of a design. With no way to shuttle data easily between disparate analog, RF, and DSP tools, there's no simple way of integrating and testing the different subsystems or predicting the performance of the system as a whole.

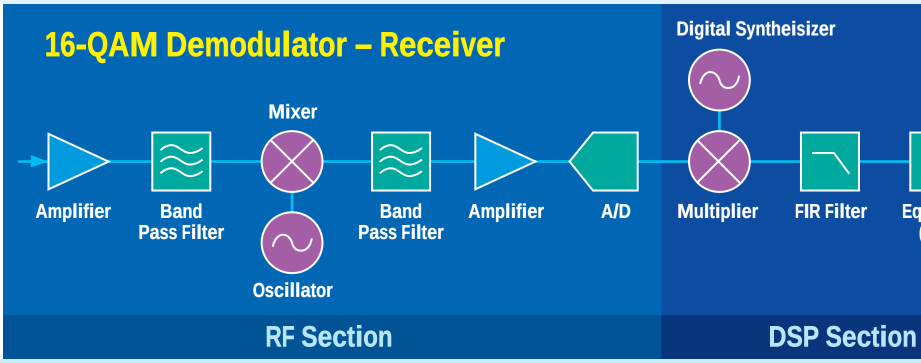
Some Typical Problems

A quadrature-amplitude-modulation (QAM) cable modem system operating in the 500-MHz band provides a good example of the problems typically faced by designers of digital communications systems and of the RF distortions they need to take into account in designing the DSP portion of such a system.

The transmitter side of the system consists of a DSP modulator and an RF section. The DSP section contains a simulated data source that feeds into a differential Gray encoder that takes a 20-MHz bit stream, performs the encoding, and maps the bits to a 16-QAM symbol constellation. (Each symbol represents a unique way of handling 4 bits of information, and the constellation is a plot of the real and imaginary portions of the signal).

The l6-QAM symbols are then fed into a filter for pulse shaping to minimize system bandwidth, frequency-shifted from baseband to a 5-MHz IF using a DSP complex-multiplication process, and then sent to an RF transmitter section that shifts the 16-QAM signal to 500 MHz and passes it through a standard sequence of mixer, filter, and amplifier.

The receiver half of the system consists of an RF section, a digital demodulator, and an equalizer. The RF receiver chain first downconverts and filters the 500-MHz signal and presents a 5-MHz 16-QAM signal to the receiver's DSP section. The DSP section then samples the signal and band-shifts it down to complex baseband. This baseband signal is then fed to a pulse-shap-



ing FIR filter to help recover the original 16-QAM symbols.

The 16-QAM symbols, which have been channel distorted, are then fed into an adaptive equalizer that continuously corrects for the frequency-response distortions caused by the reception of multiple signals with different amounts of transmission delay (time dispersion). The adaptive equalizer in this case is a decision feedback equalizer (DFE), used because the environment of this 16-QAM system is rich in these multipath transmission distortions.

To test and optimize the DFE portion of the 16-QAM receiver, it's first necessary to build up a model of the analog sections of the design in the design tool. Simulation software must allow analog-distortion effects to be added to the analysis of each portion of the design to make the entire system reflects the real-world effects the equalizer must contend with.

Dealing with Distortions

These distortion effects include phase noise of the local oscillators, power amplifier nonlinearities, circuit voltage standing wave ratios (VSWRs), and filter nonlinear phase characteristics. It is also important to include accurate modeling of the antenna and propagation channel to properly account for distortions such as frequency-selective fading and multipath distortion.

In a QAM system such as the one illustrated here, or any system that contains information in both the phase and the amplitude of the signal, it is especially important to consider multipath distortion, circuit voltage standing wave ratios (VSWRs), phase noise, and compression points on the amplifiers.

One of the most important RF distortions that needs to be considered in any communications system is multipath or propagation channel distortion that arises because of the time dispersion between reflected signals arriving at the receiver. This problem is compounded in mobile cellular systems because Doppler effects also shift the signal frequency.

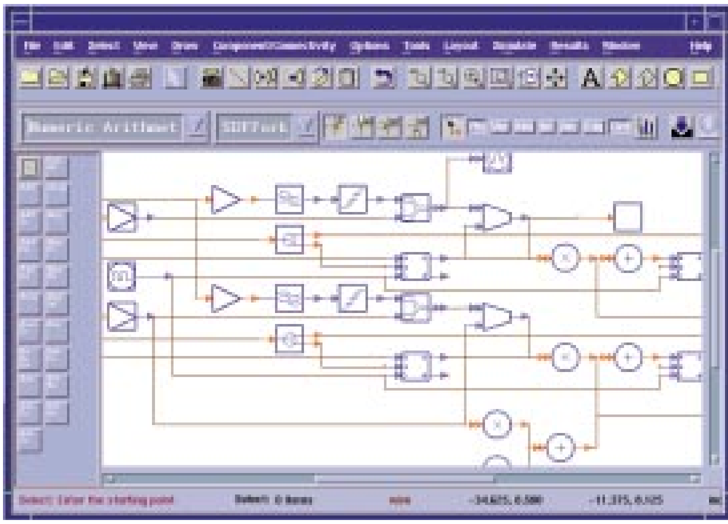
Muitipath causes inter-symbol interference and can cause your system to fail symbol error-rate requirements. As with other distortions, the equalizer in the DSP section of the receiver can correct for these time dispersion effects, but you need an adequate model of the propagation channels for simulation.

A Ptolemy Tale

Named after the second-century astronomer Claudius Ptolemaus (Ptolemy to his friends)—most famous for his earth-centered model of the universe—the Ptolemy project was started at the University of California, Berkeley in 1990. The project tackled several key aspects in the design of signal-processing and communications systems, including the design and simulation of algorithms, synthesis of hardware and software, parallelizing algorithms, and prototyping real-time systems.

The Ptolemy software environment that has evolved since 1990 provides a hierarchical, heterogeneous design framework that allows the interaction and mixing of multiple different models and simulation technologies. This is accomplished using the principles of polymorphism and information hiding (techniques well-known to anyone involved with object-oriented languages). In the Ptolemy environment, you can connect a high-level data-flow model of a signal-processing system to a hardware simulator, which you can connect, in turn, to a discrete-event model of a communication network.

The overall aim of the Ptolemy project is to develop techniques that support heterogeneous modeling, including formal "metamodels," and serve as a software laboratory for experimenting with heterogeneous modeling. The Ptolemy software developed to date has been put to work by hundreds of industry, university, and government users in applications such as signal processing, telecommunications, parallel processing, wireless communications, network design, modeling of freespace optical communication, real-time systems, and hardware/software co-design.



The newest tools let you design and co-simulate circuits with analog, RF and DSP components.

Operating at 500 MHz, the 16-QAM receiver (or any type of receiver), will suffer from reflections in the RF section because of impedance mismatches that give rise to standing waves. These standing waves produce a frequency distortion that can be compensated for by the equalizer. However, accounting for them during simulation and adjustment of the equalizer requires the simulation engine to handle a complex data type that includes the real and imaginary parts of the quadrature signal, as well as its center frequency.

The oscillators used in RF systems to downconvert the signal produce phase noise. Phase noise results from small amounts of phase modulation on the oscillator, which creates a "modulation envelope" rather than a single frequency. Excessive phase noise makes it very difficult to decode signals in communications systems that use phase to transmit data, including this 16-QAM system.

The problem could be solved with low phase-noise oscillators, but they tend to be expensive. A better approach is to simulate the system and test various assumptions about the cost of using a higher-phase-noise oscillator against the increased signal processing that may be necessary to meet error-rate requirements.

Every communications system that transmits over an RF channel uses one or more amplifiers. Unfortunately, amplifiers aren't perfectly linear and at some point will start clipping or compressing the signal. In addition to affecting the signal amplitude, these nonlinearities affect the symbol constellation, leading to errors when decoding the signal. As with other distortions, you need to include amplifier nonlinearities in simulations when designing the DSP components of a system.

With the proper RF distortions taken into account, simulations can be used to optimize the parameters of an adaptive equalizer such as the DFE (decision feedback equalizer) in the 16-QAM receiver. These parameters include the values for the step size of the equalizer coefficient calculation algorithm and the number of filter taps in the equalizer.

Tools at Hand

Fortunately, electronic design automation tools have been developed to handle the mixed time, frequency, and data-flow technologies involved in design and simulation of a communications system like a 16-QAM receiver. One tool that handles the data-flow portion of this job is called Ptolemy, which grew out of a project started at the University of California, Berkeley in 1990. The result of this work was the foundation of a software environment that allows co-simulation of RF, analog, and DSP models in a single, open design environment.

Using the results of the Ptolemy project as a starting point, Hewlett-Packard's EEsof Division developed an enhanced version, dubbed HP Ptolemy. It also developed a design suite consisting of a simulation tool, DSP Designer, and a synthesis tool, DSP Synthesizer, for optimizing high-level DSP designs simulated using DSP Designer and then implementing them as ASICs or field-programmable gate arrays.

As a result, factors such as the numeric precision of the DSP, the noise and harmonic effects of the RF down-converter, and the quantization noise of data converters in the 16-QAM system can all be accounted for at the same time. Most important, errors arising from the interaction of these effects, such as the inter-symbol interference, are revealed and can be corrected before building a hardware prototype.

For additional information on products mentioned in this article, check 1 on the reply card.

A Dynamic Duo for Simulating and Synthesizing RF/DSP Designs

HP Ptolemy, the first commercial version of the Ptolemy software developed at UC Berkeley, provides the framework for the new DSP Designer suite from Hewlett-Packard's EEsof Division.

Because Ptolemy is essentially a data-flow-based technology, EEsof made significant enhancements to it and coupled it with the capabilities of an earlier tool the company had developed. This earlier tool was based on a discrete-time technology well-suited to the modeling of RF time and frequency distortions.

In all, DSP Designer gives you the ability to generate DSP algorithms from block-diagrams; provides a comprehensive collection of DSP, digital, analog, and RF models (any behavioral model in the Ptolemy format can also be used); and uses the HP Ptolemy simulation engine to perform simultaneous data-flow simulations for DSP, harmonic-balance simulation for RF circuits, and SPICE simulation for analog circuits.

An important enhancement made to Ptolemy was the inclusion of a data type that allows the characterization of a signal using a real and an imaginary number, and a center frequency. With this data type in the simulator, it is now possible to simulate multiple RF channels and their interactions with each other.

Beyond Ptolemy

But DSP Designer is more than just an extension or enhancement of Ptolemy. It includes a real-time instrument controller, a DSP filter tool, and sophisticated post-processing capabilities.

DSP Designer's instrument controller lets you use real-world effects and distortions in your simulations and eliminates the need to hand-code some of the scripts typically needed to test hardware prototypes.

Using the instrument controller, you can replace behavioral models with actual measured data, apply measured data to a DSP simulation, export simulation data to test instruments for real-time analysis, and create arbitrarily modulated signals for evaluating prototypes.

A tune mode, part of the DSP filter tool, lets you vary digital filter parameters, as well as display and evaluate the results of multiple successive tunings. The filter tool also has a utility that generates filter coefficients from your specifications and automatically creates the filter schematic.

The data-display and post-processing capabilities built into DSP Designer also give you access to numerous mathematical functions that simplify the manipulation of simulation data. You can, for example, apply cross-correlation and FFT functions, histograms, and power spectral densities to data sets. Moreover, you can display any combination of data plots in a single window.

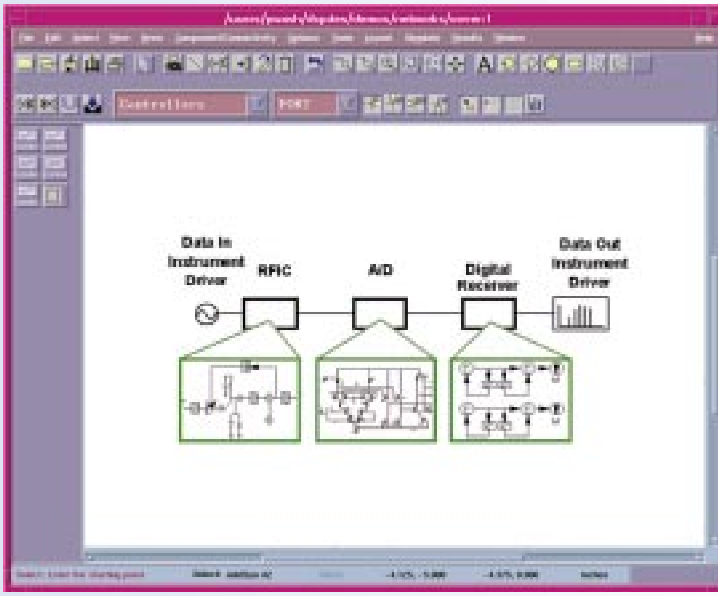
From Simulation to Silicon

Ultimately, a simulated design needs to be cast in silicon. For this task, the tool suite includes DSP Synthesis, a behavioral synthesizer developed by EEsof, that takes the high-level design created and simulated with DSP Designer and optimizes it for implementation into ASICs and FPGAs.

DSP Synthesis lets you synthesize from behavioral block diagrams down to an RTL hardware description language (Verilog or VHDL). To help ensure error-free implementations, a waveform comparator that checks for timing differences is used to validate the original simulation test vectors against the Verilog or VHDL description that has been generated.

DSP Synthesis eliminates the hand coding that conventional application-specific DSP behavioral synthesis usually requires. Instead, you specify constraints relative to performance and silicon area, and the synthesizer creates a design optimized for resource sharing, scheduling, and binding at both the DSP macro and primitive levels.

For additional information on products mentioned in this article, check 1 on the reply card.



DSP Designer lets you integrate the RF, A-D and DSP components of a system into a unified design flow