

Using behavioral models to drive RF design and verify system performance

A new generation of software tools for model-based design is enabling the creation of system-level models that system architects can use to evaluate the RF design specification. These flexible design environments allow information to flow top down and bottom up. Before, during and after the circuitry for the components and modules is being designed, the same system-level model can incorporate ever more refined performance information in order to determine the impact on system-level performance.

By Colin Warwick and Mike Mulligan

Today's wireless communications systems are built by multidisciplinary teams. The overall system behavior is the responsibility of the system architect. Implementing each part into disparate target technologies, such as real-time software, digital hardware and RF circuits, involves several specialized teams that include experts in signal processing, RF engineering and other disciplines. Smooth integration of these teams is the key to success.

In the past, the design process began with the system architecture team developing a specification that guided the work of the various design and implementation teams: embedded software, digital hardware and analog hardware. System specifications were captured as text documents. Although these often "looked good on paper" they were impossible to validate. Any flaws surfaced later in the design cycle, when running the design in various hardware and software simulators, or, worse still, at systems integration and verification. At this stage problems were prohibitively expensive to correct.

This approach, which worked acceptably well when systems were simpler, is rapidly becoming obsolete. As the complexity of RF designs increases, the complexity of specifications and the size of the teams required to build them are also increasing. For example, many designs are moving from digital signal processors (DSPs) to field

programmable gate arrays (FPGAs) and application-specific integrated circuits (ASICs) in order to use parallel processing to achieve higher bit rates. The complexity of the specifications required to define the latest generation of products is becoming difficult or impossible to capture in a text document.

The risk of producing a specification that cannot be implemented is becoming too great to ignore. For example, a 300 MHz ASIC may seem adequate "on paper." But when the design actually comes together it may provide nowhere near enough horsepower. The timing of an algorithm specified for one module may not align with the timing of an algorithm specified for another module. A successful algorithm may fail when it encounters impairments caused by real-world RF non-linearity during system integration. Since they involve system-level interaction, these types of problems are often impossible for individual design teams to detect.

Without a validated executable model, problems are often not detected until all of the different modules can be tested together at the

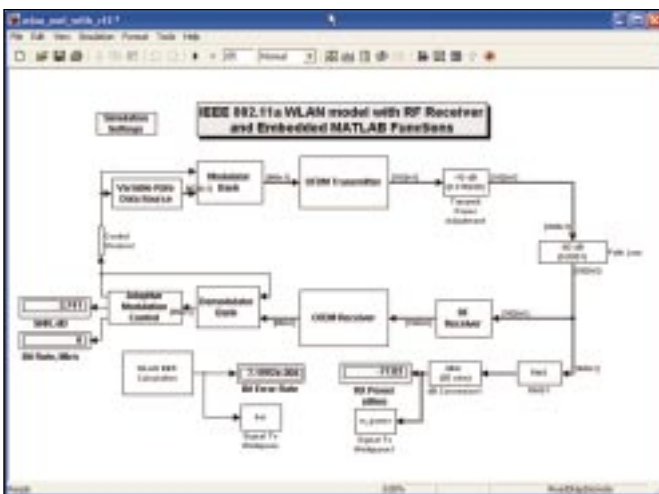


Figure 1. End-to-end model of a wireless communications link based on the IEEE 802.11a wireless LAN physical layer.

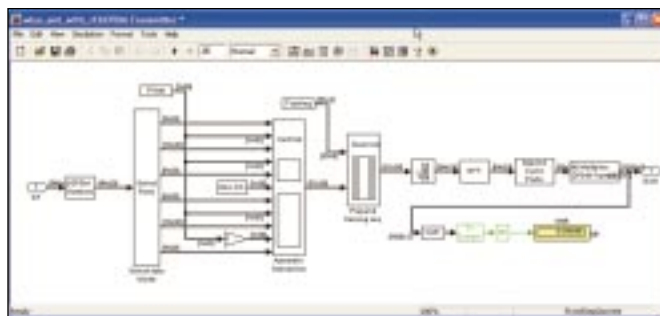


Figure 2. Under the top-level OFDM transmitter block are the signal-processing blocks required to create the waveform specified in the standard.

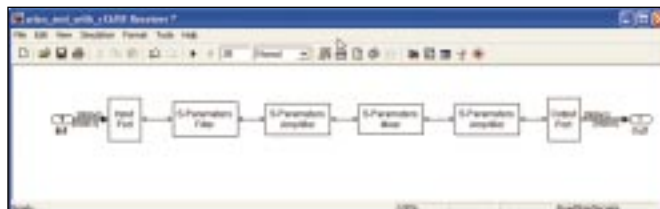


Figure 3. Under the top-level RF receiver block is a receiver model created from four component blocks and four interface blocks.

system integration stage, when a complete prototype is assembled. This approach can quickly drive up design cost because the cost of fixing a problem increases by an order of magnitude as the design progresses through each successive stage. An even more serious consequence is the opportunity cost of being late to market. Another weakness of the standard approach is that each engineering group works in isolation from the others, which makes it difficult to explore system-wide trade offs. For example, one group may have the opportunity to use a much less expensive RF component that, taken in isolation, would reduce performance slightly. What if a tiny extra expenditure on a sophisticated algorithm, say soft decision, could bring performance back into line? Without an overall system model there's no way to find the globally optimum solution.

System architects have realized that written documents are inadequate and that early system-level simulation is essential. Early attempts at simulation used a familiar and readily available tool: a procedural programming language such as C code. While C is excellent for applications programming and for programming embedded processors, systems architects found it to be a low pro-



Figure 4. The mixer block actually models a mixer and its associated local oscillator.

ductivity environment for system simulation. It lacks built-in constructs for concurrent algorithms and the real-time connection between them. This weakness is particularly obvious with frame-based multirate systems. There are several other shortcomings: It is awkward to visualize signals. It forces you to expend energy on pointers and semicolons, when you'd like to be focusing on higher-level issues. There are no supported standard libraries of signal processing workhorses such as filters, channel models, channel coding, source coding etc. Finally, it isn't easy to represent time-domain algorithms and frequency-domain RF behavior in the same model. This latter weakness and a proposed solution are the focus of this article.

Replacing text-based programming with block diagrams

Engineers continue to use procedural languages such as C or MATLAB® to simulate wireless communication systems. Such tools have so far proved successful in developing individual untimed algorithms, but don't work well as system-level design tools, because

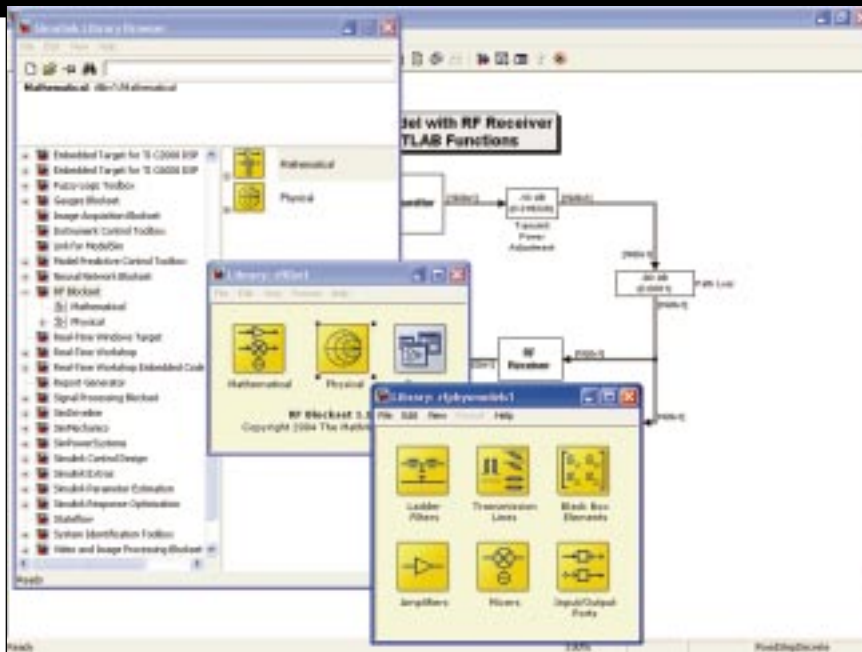


Figure 5. Blocks can be substituted with different blocks from the library as the design is refined.

concurrent algorithms joined by real-time signal flows play such a major role and procedural languages lack built-in constructs for time and concurrency. In a mobile communication system, many algorithms operate at different rates in a base station, with more operating in the handset. At the same time, the RF energy itself is bouncing off buildings and terrain, creating a dynamically fading channel. Realistically modeling the performance of an RF system under these conditions requires the ability to capture the concurrency of all of this mathematical behavior as well as the physical properties that also have a significant impact. Attempts have been made to “patch” the deficiencies. For example, SystemC provides transaction-level modeling constructs by adding a class library and a kernel to C++. However, these efforts fail to solve the basic problem, namely that it is not easy to express concurrency and real time in a text-based

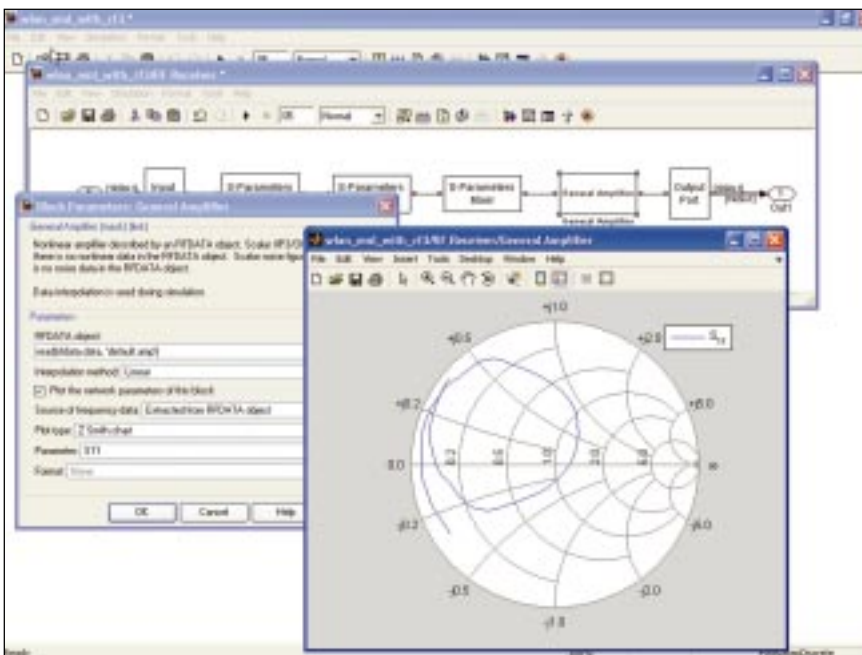


Figure 6. After the placeholder amplifier is replaced by a model specified by measurement data, the characteristics are visualized.

programming language. Customers tell us that Simulink® in their design flow is two to three times more productive than trying to coerce C code to do a task for which it wasn’t designed.

Block diagrams provide a natural solution. Block diagram environments (with their built-in scheduler or solver) are even better, because you set up each block to register its particular sample times, input frame sizes, and output frame sizes with the solver, and the solver takes care of the execution sequence. Try doing that with a “for” loop.

Model-Based Design delivers executable specification

Systems architects are beginning to use Model-Based Design to create executable specifications. These are far superior to text-based documents because they can simulate the behavior of the proposed design. We can define Model-Based design in this way: Modeling is the specification of properties of the system. Design consists of implementation tasks, such as partitioning,

encoding the desired behavior in C and hierarchical design language (HDL), synthesis, compilation, and so forth. In the document-based design era, written specifications contained flaws that emerged only during the decoupled design phase. Much expensive rework was necessary. In contrast, with Model-Based Design you build an executable model, and then base your design on it. The advantage is that an executable model can be validated, and a validated model makes the design phase much more straightforward. An ounce of prevention is worth a pound of cure.

Divide between time and frequency domain

System architects typically develop algorithms in the time domain, which is conducive to creating signal flow algorithms. In contrast, RF engineers typically talk about the frequency response of the amplifiers, filters, mixers and so forth in terms of network parameters as well as frequency-dependent noise and non-linearity. RF engineers specify hardware, design circuits, run simulations, perform physical tests, and generate mountains of this data. For example, RF engineers often work with frequency-domain simulators such as harmonic balance circuit simulators. The output of these powerful circuit-level simulators is incompatible with the baseband-complex time-domain modeling methods that system architects and signal processing engineers use. Another source of network parameter data is the measurements gathered from a network analyzer. In the “natural” frequency domain format, integration of these data into a time-domain system model for verification is difficult.

Bridging the gap between time and frequency domains

The new block diagram approach can accommodate multiple domains, including signal processing and RF. With Model-Based Design, the RF domain is specifically tuned so you can define and simulate the behavior of RF components, including filters, transmission lines, amplifiers and mixers, at the system level. Com-

ponents can be specified based on network parameters, mathematical behavior or physical properties. Creating and maintaining this domain-specific library in C code would take time away from the more productive task of using the domain for system modeling.

Use of a pre-built RF domain offers major advantages. The design space can be explored rapidly by altering parameters or algorithms. The systems architect typically begins by modeling the design with perfect RF compo-

nents and then adjusts the performance specifications of the components until performance degradation occurs. The goal is to determine what level of noise, non-linearity, and frequency domain distortions it can tolerate. The environment enables visual root cause analysis and insight. For example, network parameters can be visualized with plots and Smith® charts. In this way, the systems architect can generate the specification at a high level. These specifications guide the next

stage: detailed circuit design.

The RF engineer can pass back realistic data generated either by a circuit-level simulator or from test equipment to the systems architect in order to verify the RF design in a systems context. Again, the RF domain in Model-Based Design forms the bridge between the time domain tools used for signal processing and the frequency domain tools used by RF engineers. The conversion is more sophisticated than a simple inverse FFT from frequency domain to time domain. Real pass-band frequency responses are converted to their complex-baseband equivalent impulse responses. This allows the simulation to step forward at the symbol period, rather than being bogged down by stepping at the tiny carrier wave period. This higher throughput is valuable because a typical system simulation requires 100 million symbols to be processed in order to determine bit error rate.

In addition to the frequency response, noise and non-linearity are modeled.

Applying Model-Based Design

Figure 1 shows an end-to-end model of a wireless communications link based on the IEEE 802.11a wireless LAN physical layer. It includes the baseband and RF units of the transmitter and receiver as well as the channel model. The standard specifies the transmitter algorithm, the channel model, and the receiver performance. To implement the link, the design team must design the receiver algorithms in a way that meets the receiver performance specification.

Figure 2 shows a more detailed view of the top-level block "OFDM transmitter." At this level in the block diagram hierarchy we see the adaptive modulator block with its bank of orthogonal frequency division multiplexing (OFDM) modulators. Each is designed to operate at different data rates depending on channel conditions, as specified in the standard.

Returning to Figure 1, we see that after the OFDM transmitter block is the path loss block, which models the signal loss between the transmitter and the receiver. In this case, the wireless local area network (WLAN) channel is a static path loss, but a fading channel could easily be added if the project calls for it. Such a case might be a high mobility wireless communication project such as W-CDMA or IEEE 802.16e.

After the channel model, we have the receiver in two parts: the RF section followed by the digital signal processing section.

In this example, like most modern wireless communication systems, the RF domain co-exists with a substantial digital signal-processing domain. The signal-processing portion is realized with a mix of library blocks and user-defined blocks. In the signal-

processing section of the receiver shown in Figure 1, the equalizer and adaptive modulation control were created using the embedded MATLAB function block, one of several in the Simulink user-defined function sublibrary.

Shifting focus from the signal processing to the RF domain, the RF circuit behavior of the receiver is modeled as a cascade of four blocks, placed between an input port block and an output port block, as shown in Figure 3.

The input and output port blocks mark the transition from the signal-processing domain (where the baseband in-phase and quadrature rails are represented as real and imaginary parts of a complex time-domain signal) to the RF domain (which is specified in terms of the frequency-domain properties of components acting on a real passband signal). As mentioned previously, in the RF domain we can use a mathematical transformation to transform the frequency range of interest (a band around the center frequency) to its baseband

complex equivalent. This affords faster simulation speed.

In Figure 3, the first block after the input port is the S-parameters filter block, followed by an RF amplifier, mixer and intermediate frequency (IF) amplifier blocks. The amplifiers are defined by S-parameters, noise figure, and third-order intercept point (a simple way of expressing non-linearity). One specifies additional parameters for the mixer block: phase noise, local oscillator (LO) frequency, and so on. Because the LO is conceptually “built-in” to the mixer block, only the two remaining ports (RF and IF) are depicted in Figures 3 and 4.

These parameterized blocks serve as way to generate specifications of components yet to be built (as opposed to verifying specifications of existing components). In this sense they are placeholders for the to-be-built components.

In our example, the simulation showed that the initial concept design introduced noise and non-linearities in the front end that made it impossible to meet the 802.11a specification. The initial design provided a maximum of 12 MB/s to 18 MB/s throughput instead of the required 54 MB/s. This type of result provides the systems architect with the information that he or she needs to improve the specification. At this stage, the system architect typically runs a parameter sweep on the receiver components over a range of gains and noise figures in an effort to balance the gain produced by each stage against the noise it introduced and to develop a workable set of specifications for each component. This is the classic problem of optimizing the gain distribution. If there is too much gain early in the cascade before filtering, intermodulation products will limit performance. Too little gain early in the cascade and noise becomes the limiting factor. For each design iteration, the system architect simulated the model to calculate the receiver sensitivity, providing results that not only validated the specification but also helped to understand its sensitivity to various design parameters. In this way, the gain distribution was optimized with respect to noise and non-linearity. The system architect could at this point hand off a set of specifications for each component to the RF team.

Later in the project, the system architect may have a short list of candidate components to verify, either general-purpose or application-specific standard products from a semiconductor vendor’s catalog or custom-built circuits from the RF engineering team. In this case, the general circuit element block makes it possible to swap out one or more placeholders at a time, and import components based on power sweeps of either measured data or circuit simulator runs

(Figure 5). The modeling environment provides the ability to plot the network parameters to a polar plane or Smith chart that can be used to confirm that the correct data has been imported (as Figure 6 shows).

With this combination of real components and placeholders, we can simulate the receiver and compare its performance to an ideal gain block to verify its performance in terms of linearity, noise and network components against the system-level metrics such as throughput and bit error ratio vs. path loss.

Conclusion

This example illustrates how Model-Based Design makes it possible to develop specifications in a team environment where they can be evaluated based on cost, performance, power consumption, and other objectives. The systems architect creates and communicates the RF specification in the form of a behavioral model that can be used to validate the specification against requirements before it is turned over to the RF design team. The same model is refined during the subsequent design process to incorporate results generated by the RF engineers who are creating the actual circuit design. The RF engineers pass back behavioral information from their

EDA tools and test data to verify system-level performance early in the process.

For example, the RF engineer may not be able to achieve the specification for a certain component within the given cost constraints. The systems architect can modify the model to use an economical, achievable specification, determine the impact on the system performance, and, if necessary, make compensating adjustments to the specifications

of other components, such as using signal processing to mitigate the impairment. This process makes it possible to identify, diagnose, and correct system-level problems at a much earlier stage than is possible with a traditional development process. The net result is that problems can be identified and fixed more efficiently and system-level trade offs can be evaluated more easily in order to increase performance and reduce cost. RFD

ABOUT THE AUTHORS

Colin Warwick is RF product manager at The MathWorks. He is focused on the RF simulation domains in the context of model-based design for signal processing and communications applications. Prior to joining The MathWorks, he worked on wireless communications as a Distinguished Member of Technical Staff at Bell Labs research in Holmdel, N.J. He also designed devices for phased-array radar as senior scientific officer at the Royal Signals and Radar Establishment in Malvern, England. Warwick has authored 12 patents, and completed his bachelor, masters, and doctorate degrees in the physical sciences at the University of Oxford, England. He may be reached at Colin.Warwick@mathworks.com.

Mike Mulligan is the manager of the communications development team at The MathWorks. He has more than 20 years of industrial and academic experience. Prior to coming to The MathWorks, he worked on satellite communication system design at ViaSat, MITRE and M/A-COM Linkabit, and was on the faculty of Marquette University. Mulligan holds a Ph.D. in Electrical Engineering from Northeastern University. He may be reached at Mike.Mulligan@mathworks.com.