COMPILER
Monthly online magazine for technologists worldwide
March 2006

SYNOPSYS®

Improving System Reliability Using the Saber® Simulator in a Robust Design Flow

Michael Jensen, Technical Marketing Engineer for Synopsys' Saber Simulator Product Line, describes the attributes of a Robust Design flow, and outlines the use of Saber in improving system reliability.

Quality Versus Reliability

Quality and reliability are different, but related, measures of how well a system performs its intended task. Quality measures look at performance under nominal conditions. If, under nominal conditions, the system meets performance specifications, it is considered a quality system. Reliability measures take quality one step further. Measuring the reliability of a system looks at system quality under real operating conditions. These conditions can be internal or external to the system. It is possible to have a quality system that is not reliable. It is impossible to have a reliable system that is not also a quality system. Ensuring system reliability requires a methodical approach to system design. Using Robust Design principles in a design flow establishes this methodical approach.

Robust Design Primer

Robust design is a general but proven development philosophy focused on improving the reliability of a process or product. Improving reliability requires that Robust Design principles be an early and integral part of the development cycle. The objective is to make the end-product immune to factors that could adversely affect reliability. As shown in Figure 1, a general Robust Design methodology requires that four factors be considered in the design process: signal factors, response factors, noise factors, and control factors.

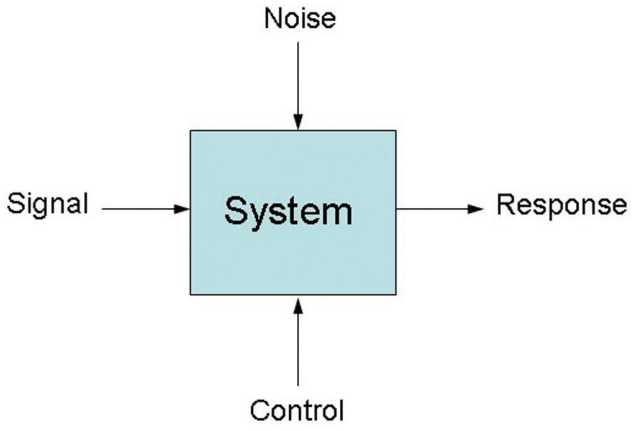

Figure 1: Factors in Robust Design

Within the context of modern systems design, these four factors have specific meanings.

Signal Factors

Signal factors are the characteristics of the system's input signal. There are a variety of factors to consider, including the signal's type (analog, digital, etc.), amplitude, frequency, spectral content, etc. The designer must understand these characteristics before an effective system design can be created. Signal factors will determine the configuration of the system's input stage, which prepares the input signal for processing by the system.

Response Factors

Response factors are the requirements placed on the system's output. Similar to signal factors, there are a variety of response factors to consider. The system must process the input signal so that the output meets performance requirements. Response factors, therefore, will determine the configuration of the system's output stage.

Noise factors

Noise factors are disturbances that cause the system's signal-response relationship to drift from nominal. These factors can be internal or external to the system. Most are beyond the direct control of the designer. Often the designer's only option for eliminating the disturbance is to incorporate noise factor compensation into the system design. To do this, the designer must first identify and quantify all noise factors that can have an adverse affect on system performance. Once the noise factors are identified and quantified, the designer must choose which factors require compensation.

Control factors

Control factors are used for noise factor compensation and are under the designer's direct control. The objective is to anticipate and compensate for the noise factors that could significantly influence the system away from nominal performance. While a noise factor may have more than one compensation solution, Robust Design principles call for the simplest, most cost effective approach. To meet this objective, the designer must often select control factors that will mitigate multiple noise factors.

Example

To illustrate how Robust Design factors apply to system design, consider a basic automotive braking system. For this example, assume the automobile will use disc brakes and that the objective is to bring its rolling motion to a complete stop.

Conceptually, the operation of a braking system is quite simple. The driver applies pressure to the brake pedal. This pressure is hydraulically or electronically transferred to the brake caliper, which forces the pads against the spinning rotor. The force of the pads on the rotor eventually slows the vehicle to a complete stop.

Referring to Figure 1, the input signal to the system is the application of pressure to the brake pedal. The main signal factor is the amount of applied pressure.

The system responds to pressure on the brake pedal by reducing the automobile's speed. The main response factor is how long it takes the vehicle to completely stop.

There are many noise factors that can affect the ability of the braking system to stop the vehicle. Common noise factors include the weight of the vehicle, the condition of the tires, the type of surface the car is driven on, the condition and temperature of the braking surfaces, and weather conditions. All of these factors are present any time a vehicle is driven. The designer must understand each of these factors and prioritize them according to their affect on braking system performance.

The designer may choose from several control factors to help compensate for braking system noise factors. Common control factors include the size of braking surfaces, computer control of the braking system, stiffness of the suspension, and adding power assist to increase the braking force. The designer must choose the combination of control factors that will best meet the system's performance specification.

Once the critical noise factors are identified and the control factors selected, a Robust Design flow is used to implement and analyze the design to ensure braking system reliability. The objective of a Robust Design flow is to meet performance requirements with the highest possible system reliability and the most reasonable system cost.

Robust Design Flow

Within the context of modern systems design, adopting Robust Design principles to improve reliability means making system performance immune to variations in design technologies, component parameters, manufacturing processes, and environmental conditions. In a Robust Design flow, these variations become the noise factors affecting system performance. The system designer must find control methods to help compensate for each variation. The method of control may be as simple as selecting high precision components or as involved as implementing new control algorithms. The matrix of possibilities becomes so complex, however, that the traditional design-prototype-test flow is not practical. Designers must move their design activities to the virtual world where powerful simulation tools, such as Synopsys' Saber simulator, support complete system design and verification using Robust Design principles.

Robust design flows are often customized according to company preference and system application. There isn't a one-size-fits-all approach. Even with customization, however, there are common elements in every Robust Design process. A complete development flow based on Robust Design techniques includes the steps shown in Figure 2.

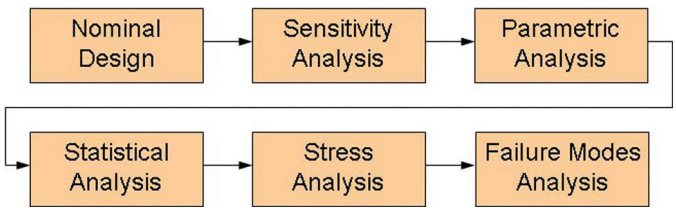

Figure 2: Steps in a Robust Design Flow

Nominal Design

The first step in a Robust Design flow is to complete the system's nominal design. The system must perform to specification under nominal conditions. The result of the nominal design becomes the response target for the remaining analysis in the Robust Design flow.

The Saber simulator supports nominal design through standard analysis (operating point, time domain, frequency domain) and a large library of behavioral and characterized simulation models.

Sensitivity Analysis

A sensitivity analysis of the system follows the nominal design stage. The designer must determine which design parameters have the most effect on system performance. The objective is to see how much system performance changes with variations in individual parameters. During a sensitivity analysis, the affect of each parameter is calculated separately. The designer analyzes the data to find which parameters most influence system performance, and then selects which parameters to focus on during the rest of the design process.

The Saber simulator supports a detailed sensitivity analysis. The designer can include every design parameter in the analysis, or specify a list of parameters that are most likely to have the biggest influence on the system's performance. Parameters are varied one at a time, and the designer can specify the amount of change.

Parametric Analysis

A parametric analysis allows a designer to fine tune the component parameters that most affect system performance. The objective is to vary specific parameters over a limited range in order to determine the set of values that best meet performance specifications. Once parameter values are selected, it's important to also verify performance over a range of environmental conditions.

The Saber simulator gives the designer access to all system parameters. Parameter values can be swept over a range in a variety of ways including linear steps, log steps, fixed steps, or through a fixed set of values. Parameter sweeps can be nested to cover all possible value combinations. Environmental parameters such as temperature can also be swept.

Statistical Analysis

A statistical analysis investigates how random combinations of parameter values can affect system performance and reliability. Parameter values are calculated based on tolerance and statistical distribution information. A series of simulations are performed, with parameter values randomly changed between each simulation run. Depending on the system, hundreds or even thousands of runs may be required to get statistically meaningful results. The results are then statistically analyzed to better understand the reliability profile of the system.

Note that statistical analysis can be extremely compute intensive. Simulating the performance of a complex system over many hundreds or thousands of runs can consume considerable computing resources. This resource requirement can be mitigated by choosing a tool that supports distributed computing.

The Saber simulator supports advanced statistical analysis. Parameter values for behavioral models can be assigned tolerances with a variety of statistical distributions, from pre-defined to user defined. Many of Saber's characterized models include tolerance and distribution information. These tolerances and distributions work with Saber's Monte Carlo analysis to provide an accurate statistical picture of the system. The Saber simulation environment supports text-based and graphics-based analysis of statistical data.

Stress Analysis

During a stress analysis, the system is simulated to see if meeting performance specifications pushes components beyond their safe operating limits. Component parameters are assigned maximum ratings, and these ratings are monitored to see if they are exceeded. When a maximum rating is exceeded, the component has been stressed. Running a stress analysis requires models that are characterized with performance rating data.

Many of the models in Saber's library either have performance ratings built-in, or allow the rating information to be added as part of the model parameterization process. Once rating information has been added, Saber's stress analysis will analyze the operational stresses placed on the model. Saber then produces a detailed report of how each component was stressed with respect to its maximum ratings.

Failure Modes Analysis

The final step in a Robust Design flow is determining how the system will perform when individual components fail. Depending on the system type and its technology, a component failure could mean a system that completely fails, a system that continues to operate but fails to meet design requirements, or a system that is able to recover from the failure and continue to meet performance specifications. Failure mode requirements are often mandated in the design specification and must be verified during the design process.

Saber's testify failure modes analyzer helps designers setup and run failure mode experiments for system designs. Parts can be caused to fail in a variety of ways and at specific times during an analysis. When a component fails, Saber can continue the simulation so the designer can investigate how the failure affects system performance.

Choosing The Right Tool

Implementing an effective and efficient Robust Design process requires simulation tools with specialized capabilities. The key tool requirements are simulation support, model library support, modeling language support, and advanced data analysis.

A Robust Design flow cannot be created with just a handful of standard analysis. A simulator must have specialized, built-in capabilities for each step of the Robust Design process: nominal design, sensitivity analysis, parametric analysis, statistical analysis, stress analysis, and failure modes analysis. Simple support for these advanced analysis is not enough. The designer must be allowed to customize the models and the analysis to meet specific system design goals.

In addition to advance analysis, a simulator must be supported by accurate model libraries. A Robust Design flow requires both behavioral and characterized device models. To ensure accuracy, the models should be based on equations that define the behavior of the device. Behavioral models should give designers easy access to key parameters. Characterized models should be created using data collected from performance bench tests, not device datasheets.

No matter how extensive the model library, there will always be situations where a needed model is not available. For this reason, a simulator used in a Robust Design flow must support standard modeling languages. The modeling language should allow the designer to create models based on actual device equations. And the modeling language should be well used and proven in the designer's industry.

Finally, a simulator must be supported by powerful post-processing tools for analyzing simulation data. The tools should give the designer insight into the details of the design, and allow the measurement, combination and transformation of design data so the designer can get a complete and accurate picture of the system's performance.

As mentioned in the Robust Design flow section above, the Saber simulator supports both the advanced analysis and model libraries needed to implement an effective and efficient Robust Design flow. Saber also supports the MAST (de-facto standard) and VHDL-AMS (IEEE standard) modeling languages which are well used and respected in the system design industry. For data analysis, the Saber design environment includes CosmosScope, a feature rich post-processing tool that gives the designer complete flexibility when analyzing design data.

Summary

Modern system design requires that reliability considerations be taken into account early in the design process. Adopting a Robust Design philosophy ensures the systematic implementation of design techniques that will lead to more reliable designs. These design techniques include: nominal design, sensitivity analysis, parametric analysis, statistical analysis, stress analysis, and failure modes analysis. Full-featured simulators, like the Saber family of design and verification tools, are required to make a Robust Design flow effective and efficient.

Web links

Reference: Introduction to Robust Design; Madhav S. Phadke
http://www.isixsigma.com/library/content/c020311a.asp

Saber: Mixed-Technology System Simulation
http://www.synopsys.com/products/mixedsignal/saber/saber.html

SYNOPSYS®

700 East Middlefield Road, Mountain View, CA 94043 T 650 584 5000 www.synopsys.com