Using ADS for Signal Integrity Optimization

White Paper

Authors:
Hermann Ruckerbauer, EyeKnowHow.de
Sanjeev Gupta, Agilent Technologies, Inc.

Characterizing a Channel: Today and Tomorrow

In the beginning there was transient simulation. Ensuring functional chip-to-chip links in a system meant performing a time-domain simulation with a SPICE simulator. The result was a time-domain waveform that was evaluated during post processing for signal integrity (SI). The main task was to measure the eye diagram, usually assuming a perfect clock as the phase reference (Figure 1). Due to rising signaling speeds and decreasing timing margins, the task was made more challenging when it became necessary to account for other effects. For chip-to-chip communication systems that used an embedded clock, the phase-locked-loop behavior was taken into account and the jitter simulated or subtracted from the remaining timing margin of the eye. For source synchronous signals, the clock was also simulated and used for setup/hold calculations or data eye generation. In addition to link architecture changes, simulation accuracy had to be improved to account for the decreased margins, resulting in complex modeling that took into account even small parasitics.

The time-domain simulation needed to include the worst case combination of all negative signaling effects. To figure out a system's real margin, the worst case intersymbol interference (ISI) had to be combined with worst case crosstalk (X-talk). The simplest way to accomplish this was to perform a simulation with a pseudorandom bit sequence (PRBS) pattern, long enough to account for the memory of the channel, and combine it with even and odd X-talk (Figure 2). While this is a simple process involving two steps—simulation and post processing—it demanded a lot of calculation time. A single simulation alone could run overnight, with the post processing occurring the next day. The huge amount of data produced made it impossible to do the post processing in real time. Although, in comparison to simulation, the time involved for post processing was short—taking just minutes to get results.

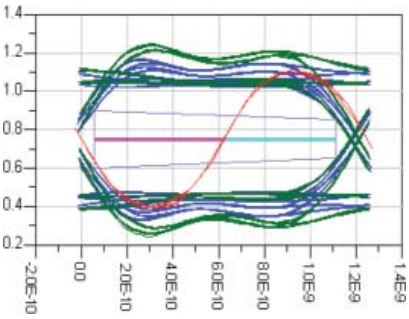

Figure 1. Data eye with phase reference and AC/DC "tSH" measurement

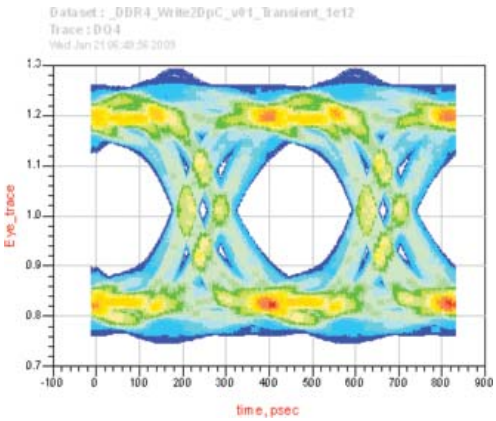

Figure 2. Time-domain simulation with a 2^{12} PRBS pattern, and even and odd X-talk

Agilent Technologies

Characterizing a Channel Today

Simulation accuracy has always been adjusted to the available calculation power. So, for optimization of a channel, several long running simulations were needed. As a result, it was necessary to reduce the accuracy and to optimize each effect separately. Even today it is a good idea to separate ISI and X-talk. It is only during the final evaluation (which is used to judge pass or fail) that all effects are taken into account in simulation. To minimize calculation time, this traditional channel characterization method has now been significantly improved.

Today, channels are characterized by a step or (im)pulse response—a method that is used by ADS Channel Simulator in statistical mode. Instead of performing a long time-domain simulation with a PRBS source, only a single rising and falling edge is simulated (Figure 3). Out of these edges, the channel behavior and worst-case data eye can be calculated. This can also be accomplished using an S-parameter characterization of the channel, but the step response has one significant advantage—it is a straight forward process to use a real driver and receiver. SPICE or IBIS models can be used for the driver and receiver, so information on the nonlinearity of these devices is already included in the information of the step responses. Most statistical eye tools also allow the engineer to use an S-parameter as the channel description, but in this case the user has to use an ideal linear (50 ohm) driver or build a behavioral model of the driver and receiver.

Statistical eye tools can calculate deterministic ISI and X-talk effects. So, at first glance, several simulations need to be done: one for the data signal (the victim), stimulating it with a rising and falling edge and observing the output; and one for each aggressor whereby the victim's response is observed. This process can be simplified by stimulating all aggressors at once and only observing the combined impact on the victim. An even simpler approach is to stimulate the step only on the victim channel and to monitor both the victim and the aggressor output, with the condition that a channel on a printed circuit board is a passive linear system. In this case, the victim's response to a stimulated aggressor is the same as the other way round. While this might not be exactly true in real-world systems, or when doing a layout accurate simulation, since the aggressor may have a different routing then the victim, as a first order result it is accurate

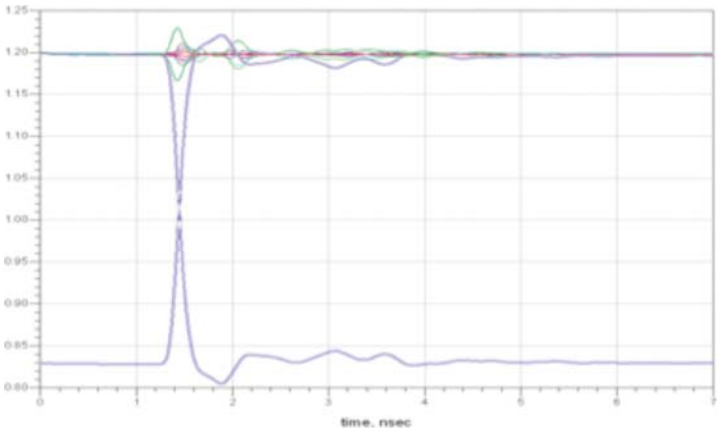

Figure 3. Simulation result for a rising and falling step. Usually real systems do not have symmetric edges so both are needed.

Characterizing a Channel Today *(continued)*

enough. The engineer can therefore, evaluate a system with multiple aggressors with a single simulation.

This approach results in short simulation times. Another big advantage of this method is that other effects can be evaluated without doing a new simulation. The statistical tools can add random jitter for Tx and Rx circuits and calculate bit error rates (BERs). Pre-emphasis and equalization can be added, and even the optimized taps can be calculated. The pattern of the signals can also be changed from a PRBS to a clock pattern. Doing a fast Fourier transform (FFT) on the signal shows the channel characteristics in the frequency domain (Figure 4).

Prior to ADS 2008, a third-party tool was required to complete this method. One could use the export from the data display to write the step response to a simple ASCII file. A better approach though, is to write it directly out from the simulation by using the "write_var" command in a "MeasEqn" or, even better, from a separate netlist. Using a netlist has a huge advantage in that the equations can be written in a normal text editor with all copy and past functions. While not intended for this function, it is possible to generate a fully automated process for data evaluation for a third-party tool using "write_var." When doing a sweep, subdirectories can be created using the "mkdir" command for each sweep step. All variables of the design can be written into a text file to document the simulation conditions. The simulation result is then exported. The script can even run the third-party tool over all the sweep cases to do the post processing and data evaluation.

With the introduction of ADS 2008, third-party tools are unnecessary due to the implementation of the timed data flow Ptolemy simulator. SI engineers might therefore prefer to perform the necessary simulations directly in ADS. In ADS 2009, a data eye can be easily generated using the Channel Simulator feature (Figure 5). For those engineers who have already implemented their own statistical tool for data evaluation, it might be advisable to stick with the external approach. But, for those engineers who are just now beginning to use this method, the Channel Simulator feature in ADS 2009 offers the ideal place to start.

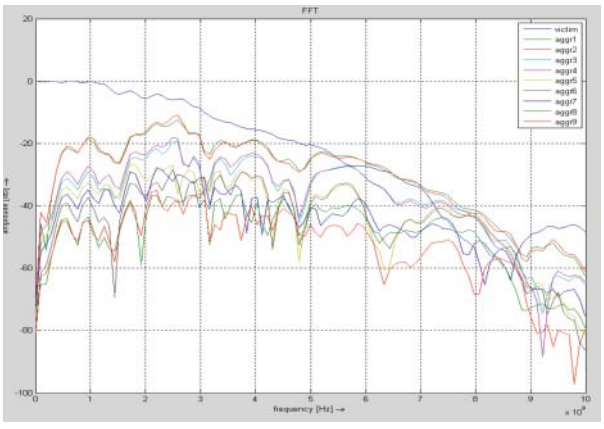

Figure 4. The FFT gives the engineer a good idea of the channel's quality (SNR)

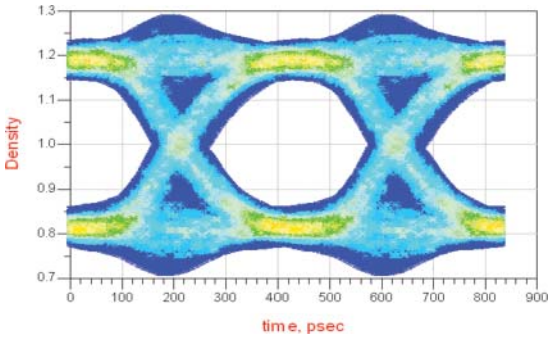

Figure 5. Data eye generated with the ADS Channel Simulator feature

Efficiently Optimizing a Channel

While there are a number of ways to investigate the quality of an existing channel, the real difficulty lies in optimizing the channel. The simplest approach is to do parameter sweeps over all interesting variables. It requires a substantial amount of work, but is a relatively straight forward process to use on a point-to-point connection. For example, on each of the parameters (e.g., board impedance (Z_0), driver impedance (R_{on}) and termination impedance (R_{TT})), the dependency can be found and the best case can be taken. In a perfect system, all impedances should match, so a configuration where $Z_0 = R_{on} = R_{TT}$ is a reasonable starting point. This is not the case in real-world systems, such as when the routing impedance is imperfect due to packaging and vias. Since all parameters interact with each other, each parameter can not be optimized on its own. Nevertheless, a first order result will provide an optimized value for each swept variable. A simulation can then be performed with the set of variables found in separate sweeps. While the result might not be optimal, it will provide a good configuration.

On a multidrop bus, channel optimization is quite complicated. As an example, consider a source synchronous, burst type, bi-directional single-ended multidrop memory bus (Figure 6). One of the most important requirements, regardless of whether we're talking about DDR2/DDR3 or the upcoming DDR4, is to connect as much memory as possible at maximum speed to one channel. Such a configuration creates a huge matrix of dependent and independent variables that must be varied to achieve an optimal configuration. Due to the asymmetric nature of the channel, the signal integrity characterization must be done twice: separately for writes and reads where some of the variables are fixed for both cases (e.g., board topology and impedance) and others can be optimized independently (e.g., dynamic ODT settings).

Unfortunately, the outcome will be not a perfect eye, but the "best" solution can be achieved by trading off the timing and voltage margins. The difficult part here is answering the question: Which is the best solution? The simplest approach to optimizing each parameter separately fails due to the high interaction between the variables (e.g., the different termination schemes that interact highly with the topology on the bus, the driver and board, and DIMM impedance). A brute force method could be used to solve the problem. It involves performing a multi-dimensional sweep and taking the best parameter combination. But, due to the large amount of possible combinations, this is not a reasonable solution.

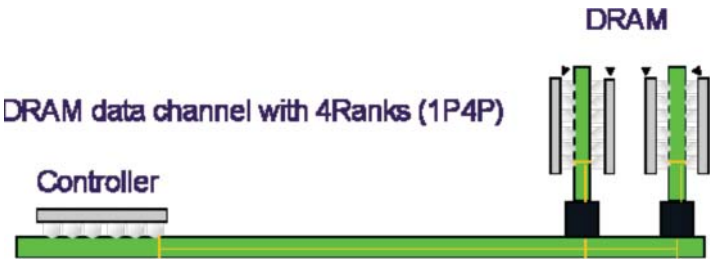

Figure 6. Multidrop memory bus example

Using ADS to Optimize a Channel

The optimizer feature in ADS offers an ideal solution to optimizing a channel, but how can it be utilized in such a complex system? One difficulty stems from the fact that the optimizer needs feedback from the simulation. An often-forgotten feature of ADS—the fact that each equation used in the data display also works in the schematic—makes this possible. In other words, once the eye calculation is proven to work in the data display, it can also be used at the schematic level. The result can be used as input for the optimizer. Unfortunately, it is not that easy to debug such a setup and even worse, a long running simulation is needed to create a data eye. The previously described method of using the step response for the eye calculation cannot be used as it requires an external tool.

Luckily, the solution to this dilemma is just a few equations or lines of AEL code away. Instead of using a step response, a unit interval (UI) wide pulse can be used as the basis for the optimization. To begin, first normalize the pulse by voltage scaling to provide a solid basis for different input signals. If this step is not taken, different combinations of R_{on}, R_{TT} and termination voltage (V_{TT}) may shift the signal and cause the equation to fail. The next step is to separate ISI optimization from X-talk. Slice the pulse waveform into pieces smaller than the UI. Where to make the cuts will depend on the real configuration. A good approach is to center the UI around the maximum point of the pulse waveform. If a clock or strobe signal is available, the engineer might also use this as the phase information for centering the UI. The outcome is now one slice with the information on the transmitted pulse (green area) and a lot of slices with the information on pre- and post-pulse disturbances (yellow area), called ISI. Subtracting each of the ISI UI's from the transmitted-pulse UI reduces the pulse like the ISI does. As shown in Figure 7, the signal is nearly settled in the 9th UI after the pulse. Therefore, simulating 10 bits should be enough for this investigation. Trying to catch the worst-case pattern with a conventional time-domain simulation would require a PRBS simulation with a 2^{10} bit PRBS pattern.

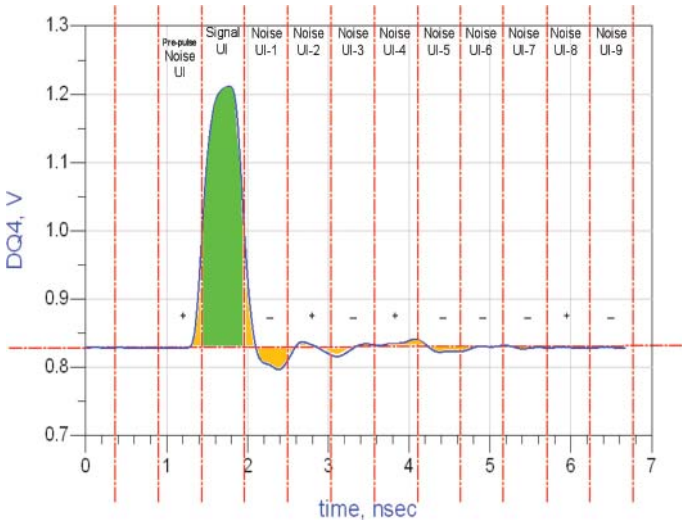

Figure 7. Pulse-response simulation result as input for the ADS optimizer

Using ADS to Optimize a Channel *(continued)*

Now, let's think about the evaluation of the data we just created. Having done the subtraction of the noise UI's from the signal UI, we can check the eye height or eye width of the remaining pulse and use the outcome as the input for the optimizer. As previously mentioned, achieving the optimal eye often requires a trade off between voltage and timing margins. Consequently, there may be two optimization goals: one for eye width and one for eye height. As it turns out, it is quite difficult to find the correct weighting to achieve the optimum result for both goals. A possible solution for this problem is to use the energy of the pulse (e.g., by measuring the area underneath the pulse), instead of using voltage and time. This can be achieved with less than 10 MeasEqn in the schematic. Using AEL code provides an even more elegant and flexible solution.

Now, since we have the ISI, let's take a look at the X-talk. As previously described, assume that we have a passive linear system and can run only a single simulation stimulating the victim with a UI wide pulse and measuring the responses on the aggressors. On the right side of Figure 8, the top of the "victim" pulse is denoted as a black dashed line. The responses on the aggressors are noted in different colors. Note that the noise of all the UI's, including the UI of the original pulse, has been calculated. This provides the necessary information for the impact from the X-talk.

In this case, the X-talk peaks are injected exactly at the crossings of the signal. If this does not fit your system (e.g., because you have a center aligned strobe), you can shift the UI borders for slicing the aggressors as needed. This action results in about 10 additional MeasEqn in the schematic, where the number of equations increases with the number of the aggressors to be considered. If the solution is implemented as AEL code, a loop can be used to create a small but very flexible and fast AEL function.

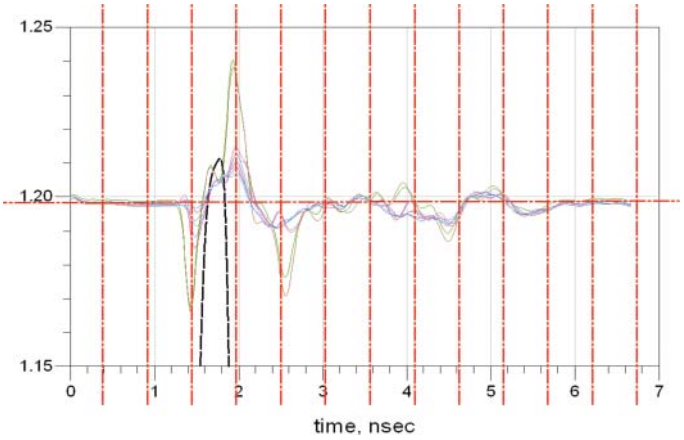

Figure 8. Single 1-pulse and X-talk on 8 aggressors

Additional ADS Features

The nice thing about this simulation is that it provides more information. Finding the worst-case pattern for a "1" is quite simple. Checking backwards for each UI to determine whether the level at the strobe point is above or below the reference level results in the worst-case pattern for the given channel. In the case shown above, this is a "0100010101**1**0" starting from UI-9 where the underscored blue "1" is the worst case pulse. The "+" and "–" signs in the noise-UI slices indicate that the resolution of the picture does not allow the engineer to read out the exact pattern. With the AEL code, there is no problem in reading out the pattern. For the worst case "0," the pattern just needs to be inverted if rising and falling edges of the pulse are symmetric.

The SI engineer can also calculate the required pattern length (PRBS or step response) for a given accuracy. For example, to get a pattern length that catches at least 97% of the ISI noise, one pulse simulation that is absolutely long enough is needed. In this case, a pattern of 10 bits would be on the safe side. There are several ways to do the calculation. Each of the implementations below is a bit correct.

One solution is to measure the area under the trace for all slices and then sum up the positive area of all noise UI's. Once you cross the 0.97 point from the overall area you know how much bits to simulate. It might be better to first use the abs() function to ensure that each part of the area inside a noise slice is accounted for and that a pulse with the same amount of positive and negative area (e.g., the noise UI-2 in the above example) is not "ignored." You might also just sum up the abs() of the levels at the sampling point. While none of these solutions will provide an absolutely correct result, they do provide just enough accuracy to accomplish the goal at hand, without requiring too much programming effort.

Conclusion

With only 10 to 20 lines of AEL code, it is possible to replace a multi-dimensional sweep of a long running PRBS time-domain simulation (including manual data evaluation) by a short, channel-pulse characterization. As shown in this white paper, there are a number of different options to implementing the required calculations. None of the options are perfect, but perfection here is not required. We simply need to find the best case configuration and a reasonable input for the optimizer. Checking the quality of the resulting parameter set is still a task for conventional simulation and the timing budget calculation. Given a practical example, this method provided a 12% improvement in eye height over the solution found with sweeps for independent, single-parameter optimization. Moreover, this method produced this result in much shorter time, further validating that by using all the ADS features SI engineers can not only improve their results, but dramatically speed up their work as well.

www.agilent.com

Agilent Email Updates

www.agilent.com/find/emailupdates
Get the latest information on the products and applications you select.

More information:
* Signal integrity solutions from Agilent EEsof EDA:
  http://www.agilent.com/find/signal-integrity
* EyeKnowHow:
  http://EyeKnowHow.de

For more information on Agilent Technologies' products, applications or services, please contact your local Agilent office. The complete list is available at:

www.agilent.com/find/contactus

Americas

Canada	(877) 894 4414
Latin America	305 269 7500
United States	(800) 829 4444

Asia Pacific

Australia	1 800 629 485
China	800 810 0189
Hong Kong	800 938 693
India	1 800 112 929
Japan	0120 (421) 345
Korea	080 769 0800
Malaysia	1 800 888 848
Singapore	1 800 375 8100
Taiwan	0800 047 866
Thailand	1 800 226 008

Europe & Middle East

Austria	43 (0) 1 360 277 1571
Belgium	32 (0) 2 404 93 40
Denmark	45 70 13 15 15
Finland	358 (0) 10 855 2100
France	0825 010 700* *0.125 €/minute
Germany	49 (0) 7031 464 6333
Ireland	1890 924 204
Israel	972-3-9288-504/544
Italy	39 02 92 60 8484
Netherlands	31 (0) 20 547 2111
Spain	34 (91) 631 3300
Sweden	0200-88 22 55
Switzerland	0800 80 53 53
United Kingdom	44 (0) 118 9276201

Other European Countries:
www.agilent.com/find/contactus

Revised: October 1, 2009

Product specifications and descriptions in this document subject to change without notice.

Printed in USA, October 19, 2009
5990-4783EN

Agilent Technologies