An IBIS-AMI Simulator for the rest of us

David Banas - July 11, 2017

Perhaps, you've heard of *IBIS-AMI* (algorithmic modeling interface) and thought, "Hey, I should learn all about this." But, when you asked around, you found that your company didn't have a license for any of the commercially available IBIS-AMI simulators. You may have even made some initial inquiries of vendors only to find that those licenses are rather expensive. And at that point you may have concluded that it just wasn't in the cards that you would learn about IBIS-AMI modeling. Well, don't give up hope just yet, because have I got a deal for you! How much would you pay for an IBIS-AMI compliant simulator, which was not only free of any licensing costs, but also made its source code freely available to you so you could study/modify it, as you pleased? But, wait, don't answer yet. Because, if you go to the *Instant Gratification* page of the *PyBERT* Wiki, right now, you can be up and running with just such a simulator in less than 30 minutes. Now, how much would you pay? But wait, don't answer yet. Because, if you… (Actually, go ahead and answer, because I can't find Ginsu steak knives anywhere.)

A brief history of IBIS-AMI

The IBIS-AMI modeling standard is an extension to IBIS, which predates it by many years. It was intended to provide the same sort of *behavioral* approach to high speed serial link modeling that IBIS brought to parallel I/O modeling. IBIS was chosen to host this new algorithmic approach to serial link modeling, for several reasons:

- The metaphor maps very nicely; in both cases the intent is to abandon a circuit level description and instead embrace a behavioral approach. This shift of metaphor has several advantages, including:
  - faster simulations, and
  - potential IP protection for silicon/model makers.
- IBIS has been very successful at obviating the need for PCB designers to learn SPICE in order to get their jobs done. The hope was that, in a similar vein, IBIS-AMI would obviate the need for high speed serial communication link designers to learn MATLAB – or any of the other equivalent numerical modeling platforms – which have historically been used for this purpose.

Indeed, with an IBIS-AMI compliant simulator in hand, as well as the necessary IBIS-AMI models, a system designer is now able to produce a production worthy high speed serial communication link design, without having to understand the details of *channel equalization*. And that has been a very empowering paradigm shift for the telecommunications industry. All that remains is to get this new capability into the hands of more designers, not just the privileged few, whom happen to work for a company willing to shell out $40,000 for a license to one of the commercial tools. Enter *PyBERT*.

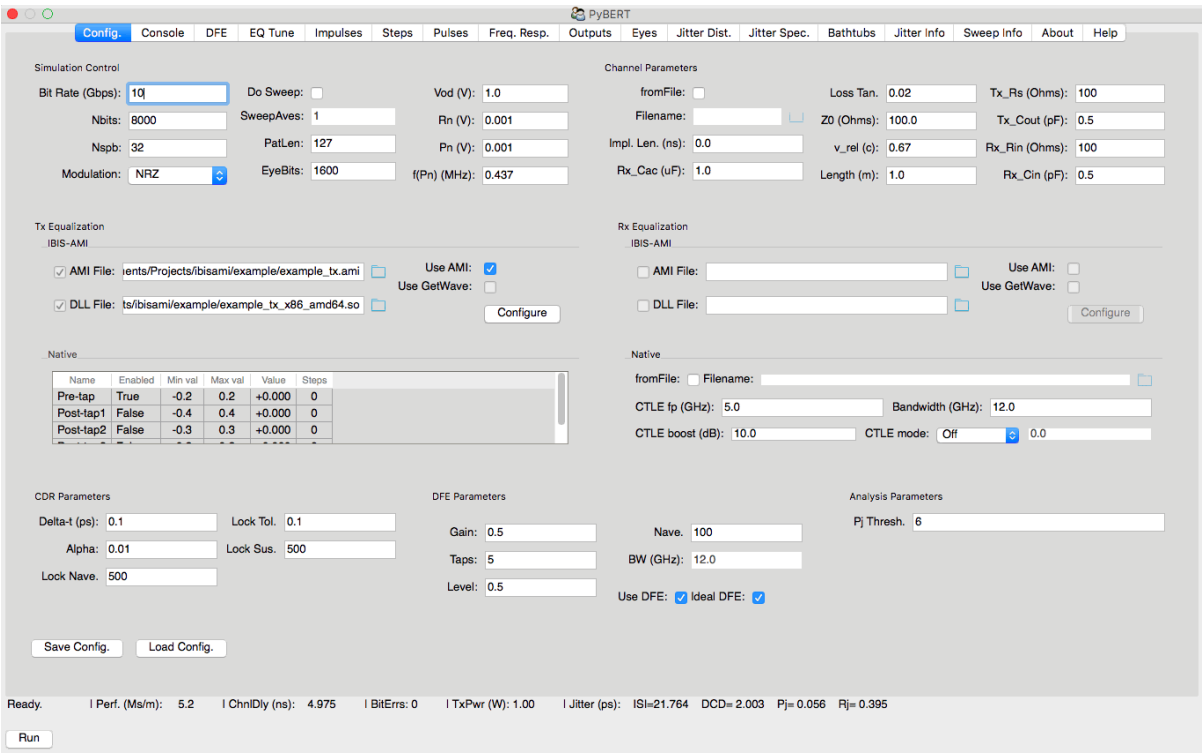
A brief introduction to *PyBERT*

PyBERT was originally just a learning tool intended to help the curious telecommunications engineer better understand the wizardly topic of *channel equalization*. When I first started writing PyBERT (ca. 2014), the topic of channel equalization was still considered to be a "black art". And those who had demonstrated some proficiency at it enjoyed a certain wizardly status, and were profiting from that status handsomely. Worse, some of these wizards were actively discouraging other engineers from developing their own understanding of this topic, hoping no doubt to extend the lifespan of this new racket they'd discovered and milk it for all it was worth. This really bothered me, because I understood channel equalization and knew that the mathematics behind it weren't really all that mysterious. I really wanted to pull the curtain aside on these self-proclaimed wizards and show people that the wizards' machine was operable by them too. So, I wrote PyBERT, intending for it to function as a community owned and maintained resource for the curious telecommunications engineer interested in learning about channel equalization. Last year, over the holiday break, I decided to give PyBERT the ability to use IBIS-AMI models – as an alternative to its own native algorithms – for modeling the effects of channel equalization.

Your first PyBERT IBIS-AMI simulation

Getting up and running with PyBERT is easy. Just go to the *Instant Gratification* page of the PyBERT Wiki and follow the instructions there. If you bump into any trouble, I'll be happy to help you navigate around it.

Running your first IBIS-AMI simulation in PyBERT is pretty straightforward. Just click on the *Config*. tab and start working your way from the top-left section (*Simulation Control*) to the right (*Channel Parameters*) and then down to the *Tx_Equalization* and *Rx_Equalization* sections.

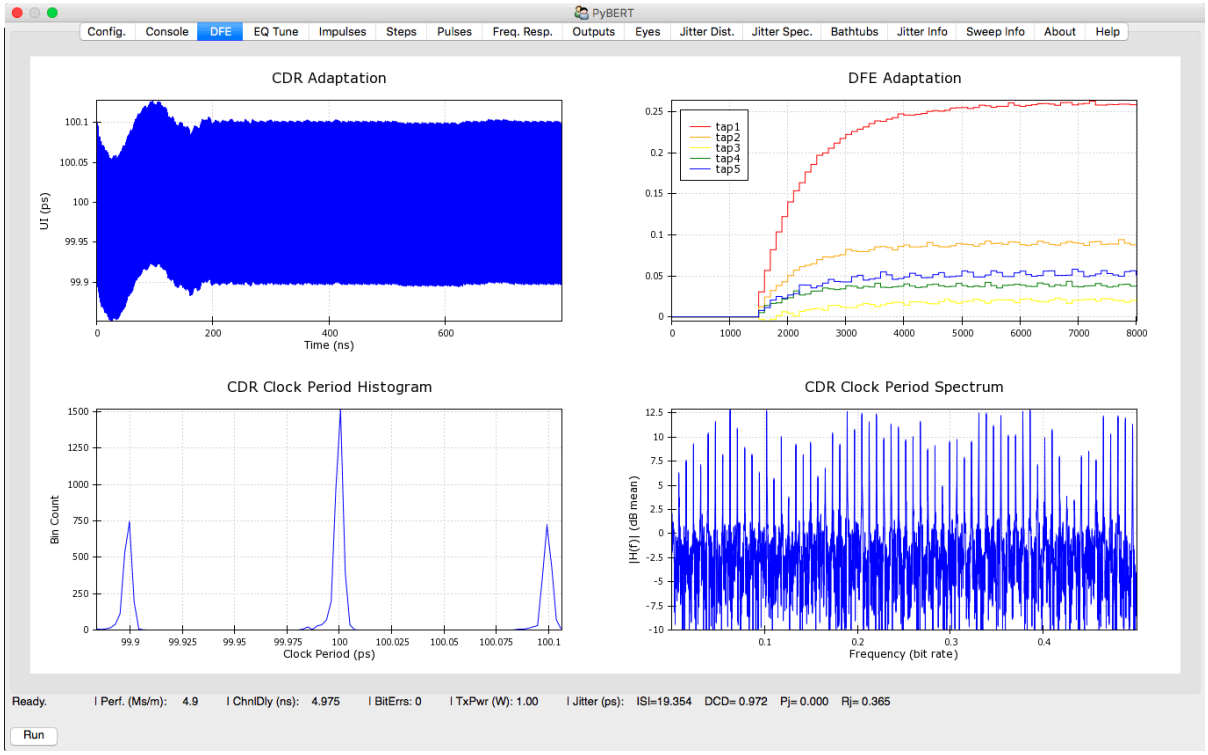


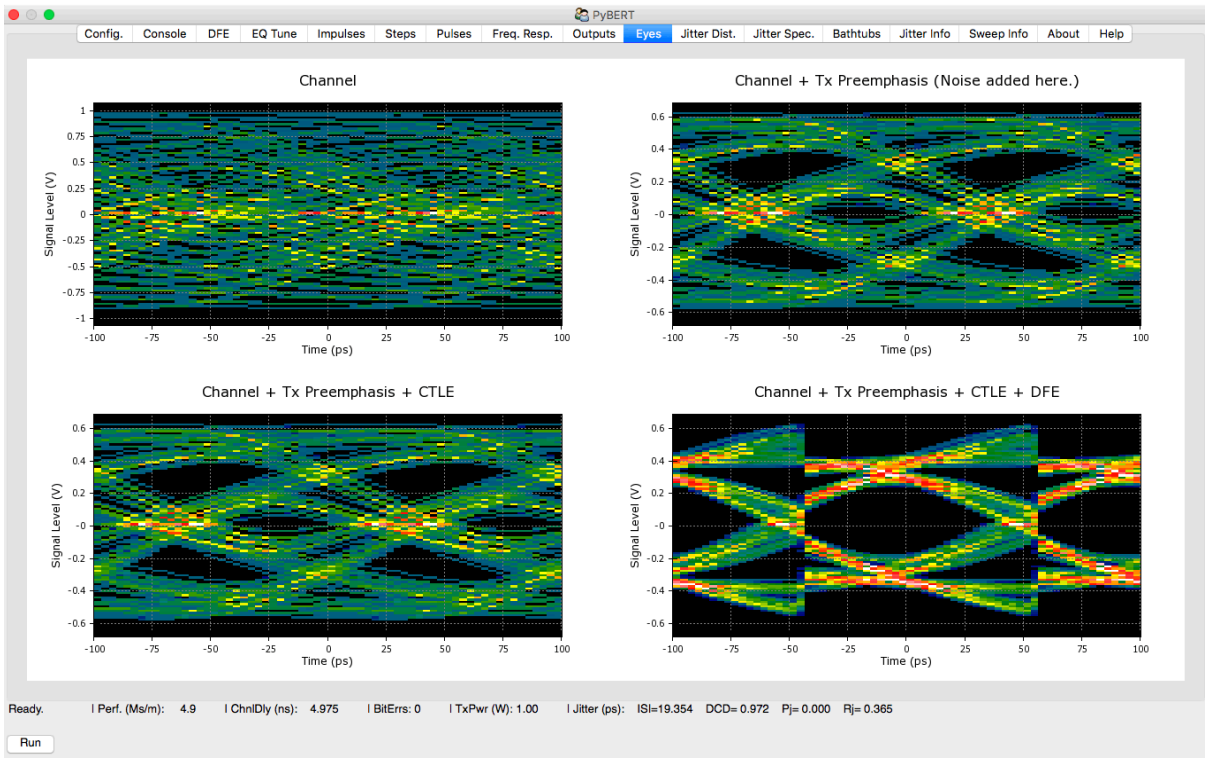
You'll notice that both of these contain two subsections: *IBIS-AMI* and *Native*. Native equalization modeling is the default in PyBERT. You'll need to explicitly activate IBIS-AMI mode, by:

1. Selecting a DLL file.
2. Selecting an AMI file.
3. Clicking on the Use AMI checkbox.

Decide whether you want to run the simulation in Init()-only or GetWave() mode, and check, or leave unchecked, the *Use GetWave* checkbox accordingly. Finally, click the *Configure* button, to set up each of your AMI models. When you're done, click the *Run* button at the bottom of the PyBERT main window. (You can safely ignore the options in the following three sections, all located in the bottom row of the *Config*. tab: *CDR Parameters*, *DFE Parameters*, and *Analysis parameters*.)

You should see the status (located near the bottom-left corner of the PyBERT main window) change state several times. And, when it returns to "Ready", your simulation has completed and you are ready to view the results. You do this, by clicking on the other tabs of the PyBERT main window, generally working your way from left to right, depending upon what you're most interested in seeing.





Some (hopefully) helpful tips:

- The IBIS-AMI mode of PyBERT has not been thoroughly tested in either the Duo-binary, or PAM-4, modulation mode. For now, only NRZ is recommended.
- If you choose to load your channel response from a file, you may use either an impulse or a step response. PyBERT will automatically figure out which you have given it.
- If the meaning and/or intent of a particular configuration item is unclear, hover your pointer over the data entry field for that item for a few seconds, and a helpful tip will pop up explaining the purpose of that item.
- The checkboxes to the immediate left of the “AMI File” and “DLL File” labels are not user selectable. They appear checked, when PyBERT has validated your selected file. If you select a file and the associated checkbox remains unchecked, something is wrong with your file (at least, from PyBERT’s perspective). If this happens, and you’re sure you have a valid file, please send the file in question to me (capn.freako AT gmail.com) and I will debug the issue.

That’s about all there is to running an IBIS-AMI simulation in PyBERT.

Also see:

- **Free yourself from IBIS-AMI models with PyBERT**
- **IBIS-AMI for rocket scientists**
- **AMI models: What, why and how?**
- **SiSoft publishes a resource guide for IBIS-AMI model development.**
- **What you need to know about IBIS 6.1**

Some helpful resources

Following are some resources you can use, to learn more about PyBERT and IBIS-AMI modeling:

- PyBERT Wiki – to view quick start instructions, FAQ, etc.
- PyBERT Main Github Page – to download, clone, or fork the PyBERT source code.
- PyBERT Mailing List – You can ask questions of the general PyBERT user community, by sending e-mail to: pybert AT freelists.org.
- PyIBIS-AMI Wiki – If you're interested in learning about the underlying machinery that made it possible to give PyBERT IBIS-AMI capability in the time span of a holiday break.
- IBIS Home Page
- IBIS-ATM Workgroup – The AMI portion of the IBIS standard is shepherded by the IBIS Advanced Technology Modeling working group.
- IBIS-ATM Mailing List Archives
- Python Home Page – Learn about Python, the language used to write PyBERT.
- NumPy/SciPy Libraries – Learn about the numerical computing libraries which make tools like PyBERT possible to develop in Python.

—David Banas is Principal SERDES Applications Engineer at eASIC.