

INSIDERS' GUIDE: FPGAs, TOOLS, AND BOARDS



FEATURED INTERVIEW:

EXCERPTED FROM WWW.EG3.COM



Prepared by:

eg3.com

Jason McDonald, Senior Editor

eg3.com

tel: 510.713.2150

email: info@eg3.com

web: <http://www.eg3.com>



MATHWORKS: MATLAB AND SIMULINK FOR FPGA DEVELOPMENT

24 October 2008: MATLAB and Simulink for FPGA Development

INTERVIEWEE. KEN KARNOFSKY
 MARKETING DIRECTOR OF SIGNAL PROCESSING AND COMMUNICATIONS
 TEL. 508-647-7000
 EMAIL. sales@mathworks.com
 COMPANY. THE MATHWORKS
 WEB. <http://www.mathworks.com/>

Q. First of all, tell us a little bit about yourself and your responsibilities at the MathWorks.

A. As the marketing director of signal processing and communications for The MathWorks, my role is managing a team that is responsible for product management and marketing for a range of products, including algorithm development, design, and simulation as well as implementation and verification workflows for FPGA and ASIC hardware and DSP software. I also work closely with MathWorks partners, and participate on the strategy committee of the Design Automation Conference.

Before joining The MathWorks, I have been involved in research, development, and marketing of advanced signal processing and data analysis technologies, which have helped me understand customer challenges and anticipate industry trends.

Q. Obviously our FPGA guide concerns FPGAs and FPGA-based designs. Yet MathWorks is not always the first company that pops into mind when it comes to FPGAs, so please explain the product groupings that you offer that are most appropriate to the FPGA community.

A. We're well known in the embedded systems design community and we've been supporting FPGA design workflows since 2000, when we collaborated with Xilinx on their introduction of their *System Generator* product. In 2003, we announced co-simulation support for Mentor Graphics *ModelSim*. Today, we offer many more products that connect *MATLAB* and *Simulink* to FPGA design workflow as well as strong partnerships with both Xilinx and Altera.

On the verification side, our EDA Simulator Link products allow engineers to co-simulate their Verilog and VHDL code running on Mentor Graphics *ModelSim*, Cadence *Incisive*, or Synopsys *Discovery* with *MATLAB* and *Simulink*.

On the design implementation side, we offer *Filter Design HDL Coder* and *Simulink HDL Coder*. These products generate synthesizable Verilog and VHDL code to enable rapid implementation of *MATLAB* and *Simulink* based algorithms on to FPGAs.

In addition to the implementation code, the *HDL* coder products generate test-benches, simulation scripts, and synthesis scripts for invoking downstream FPGA design tools. Both of the code-generation products also take advantage of the co-simulation capability offered by the EDA Simulator Links. *Simulink* HDL Coder also makes use of these co-simulation blocks to enable 'black-box IP' integration of hand-written HDL code with the automatically generated HDL code.

Q. Where is the unique value that MathWorks brings? What sorts of applications or design challenges do you feel are most likely to benefit from bringing MathWorks' tools into the design process?

A. Our FPGA design and verification tools are bridging the system design to FPGA implementation gap. These tools are enabling rapid prototyping of complex Signal Processing, Communications, and Image Processing algorithms onto FPGAs.

For example, many companies that create algorithmic IP for applications such as signal processing typically design those algorithms in *MATLAB* or *Simulink*. When implemented in silicon, these algorithms may constitute 50% or higher amount of the total gate count. The complexity of these algorithms is forcing engineers to spend more time at the system design stage with less time available for chip design and verification. At the same time, since many design engineers are working with draft standards, they constantly have to react to changing specifications. Our workflow and products are ideal in this kind of rapidly changing design and implementation environment.

Q. What is the philosophy behind “model based design?” Is there a philosophical mind shift required to use MathWorks products for FPGAs? Does it require engineers to rethink? Or are there transitions between doing things more “by hand” and doing them by “model?”

A. The philosophy of Model-Based Design is a simple one. We call it Model-Based Design because we put the model at the center of the workflow. In Model-Based Design, the model includes the device or algorithm that is being designed, the environment it will operate in, the input stimulus, and the output measurement metrics, scopes, and instrumentation. The model becomes an executable specification that unambiguously describes system behavior. Through simulation, you can validate that the modeled behavior meets system requirements.

For example, consider that you are designing a wireless communication system using Model-Based Design. In addition to the digital receiver and transmitter sections, your Simulink model would include the RF and analog models alongside the channel impairments you are expecting. So, even though you are only implementing the digital section in the FPGA, you are able to simulate the digital sections with the analog, RF, and channel impairments to ensure that your algorithm is robust enough to deal with the real world environments. Algorithm component models can be developed in Simulink, MATLAB, or C code and integrated into the system-level Simulink model.

Since we provide libraries of many of these system level models, digital design engineers don't have to piece-meal their own models of what an analog, RF, or channel model component would look like. When simulating in *Simulink*, for example, they can easily view the numerical as well as visual results via the different scopes we provide.

So, now going back to the top, once you have modeled your algorithm to your satisfaction, we provide tools to translate that model into a fixed-point model, which is necessary for efficient hardware or software implementation. In the manual workflow, this can be a time-consuming effort. If you use too few bits, you could end up with overflow conditions. If you use too many bits, you could be wasting silicon and burning excess power.

From this fixed-point model, you are then ready to automatically generate 'C' or HDL code. Using our targets and links products, you can then connect directly to downstream tools offered by our partners in the Embedded System design community or EDA industry. This is particularly important in a world where FPGAs are often used for embedded co-processing along with DSPs and microprocessors.

Model-Based Design emphasizes system level design and uses design automation technologies to accelerate the implementation of the system you are trying to create. For FPGA designers, Model-Based Design is not new. It is a take on the top-down design workflow. As you know,

because of large design sizes, chip design engineers adopted bottom-up design workflow, which allowed engineers to build and simulate one block at a time and then stitch the 'core' together later. There are too many issues with bottom-up design workflow to list here.

But, everyone knows the advantages of top-down design. Since Model-Based Design allows high level of design abstraction, engineers can do 'block-level' IP creation using a 'checker-boarding' approach in a Simulink model. Since the Simulink model is the test-bench as well as the IP creation 'work-bench', it becomes the central part of any design team. Engineers in Bangalore and San Jose will swap in their HDL IP into the same Simulink model and ensure that it works correctly.

Q. On the language side, how extensive is the support for VHDL and Verilog? Are MATLAB and Simulink new ways to create better VHDL and Verilog? How so? What are the design trade offs?

A. *MATLAB* and *Simulink* are definitely the right place to start for system level design of signal processing, communications, control, and other complex applications. This level of abstraction works really well for rapid evaluation of your algorithm since you can evaluate many architectures without committing yourself to hundreds or thousands of lines of code. In a nutshell, you benefit several ways:

- 1) Dedicate more time to evaluate your algorithm and architecture to achieve the right balance of area/speed/power trade-offs
- 2) Automatically generate implementation code ('C' or HDL) to quickly prototype your design on actual hardware
- 3) Use the *Simulink* or *MATLAB* model as a test harness to validate your implementation in the context of the entire system

Our HDL code generation products generate IEEE 1364-2001 compliant Verilog and IEEE 1076 compliant VHDL code.

On design tradeoffs, we are often asked about efficiency compared to manually coded designs. On this topic, you have to answer these questions: "When comparing to automatically generated code, which iteration of the hand-written code are you comparing to? How long did it take to arrive at that final optimal iteration?" As you know, the first iteration of the hand-written code is typically coded just to achieve functionality and an iterative process with synthesis tools is then used to refine the hardware implementation to achieve the area/power/speed trade-off.

From experience we know that design specifications almost always change in the middle of a project. If you're operating at the Verilog or VHDL code level, you may have a significant amount of work to do to make the necessary changes. On the other hand, if you're operating at a system level, you can iterate faster, verify that they work properly, and even accommodate changes in the target hardware architecture.

In hardware design, there is no silver bullet. So, we provide multiple architectures for many of our blocks. For example, engineers can choose to implement our 'product of elements' block as a serial, cascade, and parallel design implementation. They can choose to unroll loops. They can specify design latency to help the design synthesis process.

Our customers are telling us that the quality, readability, and efficiency of our code is similar to what they would do by hand.

Q. On the design tools front, there are of course many "free" tools from Xilinx and Altera. How do the MathWorks' products interface with the "free" tools? What additional value do they bring to the "free" tools?

A. It's important to note that we have a constructive working relationship with both Xilinx and Altera, and that their DSP tools are based on *Simulink*. On our web-site we offer a workflow

paper, for example, that shows how you can use *Simulink* HDL Coder with Xilinx System Generator in the same design.

That said, our HDL code generation products offer some key advantages. First consider the workflow. The 'added value' of our products, as you said, is that the customer uses native *Simulink* blocks for design, simulation and code generation. The vendor products provide implementations that work well on their devices, but they require you to re-design the *Simulink* model using their proprietary Blockset. As design engineers we know that these design translations can be very time consuming and error-prone. Once you make this translation, you commit yourself to working at this lower level of abstraction (hardware components). So, design iterations become more difficult.

Second, the HDL code we generate is highly readable and portable. The design can be targeted easily to ASICs or FPGAs and HDL synthesis tools from FPGA and EDA vendors, the code is very similar to what you would write by hand. Our customers tell us that code readability is important to them when they need to do system level debug and coordinate the effort of multiple engineers working on a design. While the *Simulink* blocks are typically used for datapath designs, we also offer the ability to include finite state machine logic using our Stateflow product, as well as custom-authored IP using Embedded *MATLAB* functions in the Simulink models.

Q. Similarly, Mentor Graphics and others offer third party tools for FPGAs. What is the relationship of your products to those of Mentor Graphics and others? Are they competitive, complementary, both?

A. The RTL-level EDA simulation and synthesis tools are complementary. In the area of system level design, we believe that Model-Based Design offers the key advantage of high-level design space exploration that other products in the EDA space don't offer. For example:

Simulink is a multi-domain modeling and simulation environment. For electronics engineers, this means that they can do analog, mixed-signal, RF, channel modeling, and digital design in one integrated environment. The simulation, evaluation, and visualization of results take place within Simulink using the many scopes and meters we provide.

The established EDA workflow, on the other hand, is very fragmented. One has to do digital design in one tool-set and the analog/mixed-signal design using another tool-set. Visualization and evaluation of the results is then cobbled together via scripts or custom made tools.

In the area of chip-design, we offer complementary workflow with our EDA partners. For example, our *EDA Simulator Links* connect *MATLAB* and *Simulink* to the HDL simulators including Mentor Graphics ModelSim and Questa, Cadence Incisive, and Synopsys Discovery.

Q. Finally, tell us a little bit about pricing packages and about where to get more information. Are there webinars, seminars, and other "learning" venues? Are there demos or evaluation versions of the tools?

A. The *EDA Simulator Links* and *Filter Design HDL Coder* are priced at US\$2000 for perpetual individual license. *Simulink HDL Coder* is priced at US\$15,000 for perpetual individual license.

More information, including detailed product descriptions, access to demos and webinars as well as support and training resources are available at:

- EDA Simulator Link MQ 2.5 for Mentor Graphics ModelSim:
<http://www.mathworks.com/products/modelsim/index.html>
- EDA Simulator Link DS 2.0 for Synopsys Discovery:
<http://www.mathworks.com/products/linkds/index.html>

- EDA Simulator Link DS 2.0 for Synopsys Discovery:
<http://www.mathworks.com/products/linkds/description1.html>
- Filter Design HDL Coder:
<http://www.mathworks.com/products/filterhdl/index.html>

We also offer access to other prerecorded webinars such as:

- [Rapid FPGA Implementations with Model-Based Design](#)
- [HDL Functional Verification with MATLAB and Simulink](#)

Additionally, evaluation versions of these products can be made available by contacting sales@mathworks.com.

Q. Thank you for this interview.