

Debug systems on your desktop—not in the lab. It's the only way to meet the architectural and algorithm requirements of multistandard communication SoCs.

Designing Multistandard Communication Devices

By Andrea Kroll and Johannes Stahl

Debug systems on your desktop—not in the lab. It's the only way to meet the architectural and algorithm requirements of multistandard communication SoCs. Modern, multistandard communication products cannot be designed without working at the system level and without using the right modeling and simulation tools. Only in that way can you satisfy your algorithmic and architectural requirements. Only in that way can you achieve the flexibility of modeling and simulation speed you need to conduct fast design iterations.

For algorithms, you need to figure out whether the basic requirements and layer 1 protocols are met. More importantly, you must optimize the amount of resources needed for implementation, specifically, fixed-point arithmetic implemented in hardware.

Multiprocessor architectures for communication systems need to adhere to strict real-time constraints in order to meet layer 2 and up software protocols. Moreover, there are product throughput requirements that differentiate a final product from its competition. Determining those requirements early in the design cycle is important because it will influence HW/SW partitioning or

even the selection of the right processor cores.

Here's the background

Modern consumer products exhibit very advanced communication features. If the remote control for a TV represents consumer device “communication” decades ago, then a multiport, wireless residential gateway is what you see today. Cost requirements for consumer products typically require specialized chips containing one or more processor cores.

Good examples of such system-on-chips (SoCs) are products based on TI's OMAP platform, in which a DSP and a RISC processor are inte-

grated and enhanced with application-specific hardware.

Making sure that such SoCs perform according to the requirements specification involves dealing with all aspects of the system specification very early in the design process. Design and product managers want to know that the system specification is correct before they give the go-ahead for SW engineers and HW designers to implement the application software and the chip.

Designing such a system sequentially, namely building chips and boards first and writing application software and then experimenting in the lab is no longer an option. No one can afford to risk an extra \$500,000 and more for a chip re-spin, if the design has specification bugs.

Is there a problem?

Let's take the example of a residential gateway, one that receives Internet traffic through ADSL and routes it through 802.11 WLAN outputs. The system level design challenges for such a consumer product are immense. They range from the design of the 802.11 transceivers to

the optimization of the routing from the ADSL to the 802.11 outputs.

Are there any problems with a particular design? With growing complexity the answer is becoming harder and harder to establish. It is becoming impossible to answer system performance or system compliance questions with low-level representations of the system, such as using Register-Transfer-Level (RTL) descriptions of the hardware.

Consider the algorithmic design of such a system. For a long time, individual algorithms have been investigated by computer simulation or in terms of their response to certain test sequences. Today it is no longer sufficient to treat a filter algorithm in isolation; you must consider a complete sequence of processing functions—such as framing, channel encoding, modulation, transmission, demodulation, synchronization, channel decoding and de-framing—to find out, whether an algorithm in this chain performs as expected.

Of particular challenge is the correct and efficient modeling of the physical transmission media and the analog signal processing in this chain. Without proper models for those components, the digital signal processing evaluation becomes entirely meaningless. Besides the modeling know-how of the system designer, the capabilities of a tool such as System Studio will

determine success. Optimized static scheduling and execution of multi-rate models, which are used to model the system behavior, are important to end up with the necessary high simulation speed.

Designing analog/digital systems with enough margins based on the imperfections of the analog components belongs to the “fine art” of

algorithm optimization, which can only be done with a combination of designer experience and simulation tool.

Given the project pressure, the system designer may treat simulation speed for modeling accuracy, if the tool has limitations. Insufficient modeling of an oscillator can cause the receiver SoC to never acquire

frequency lock during system integration in the lab. Imagine the project manager coming into your lab and asking: “Is there a problem?”

SoC architectures are evenly challenging. In many cases, designers will have already picked a certain RISC processor, DSP and bus, such as AMBA or OCP, to implement their desired algorithms and protocols in SW.

Typically, the decision is more driven from a commercial, rather than technical, aspect. Yet, it needs to be proven, that the overall requirements can be met. The typical parameters which designers need to optimize are memory sizes, memory system architectures, number of buses, interrupt priorities, and processor balancing.

The problem with these aspects is the complex interaction of hardware and software components. Even if the system architect tries to estimate bus loading and memory sizes using static analysis on a spreadsheet, it will not give enough security that the complex dynamic situations are

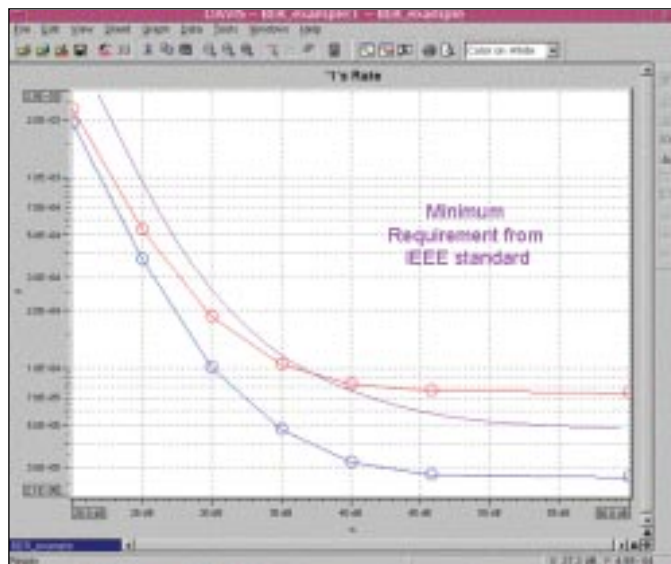


Figure 1: BER Performance Graphs in System Studio

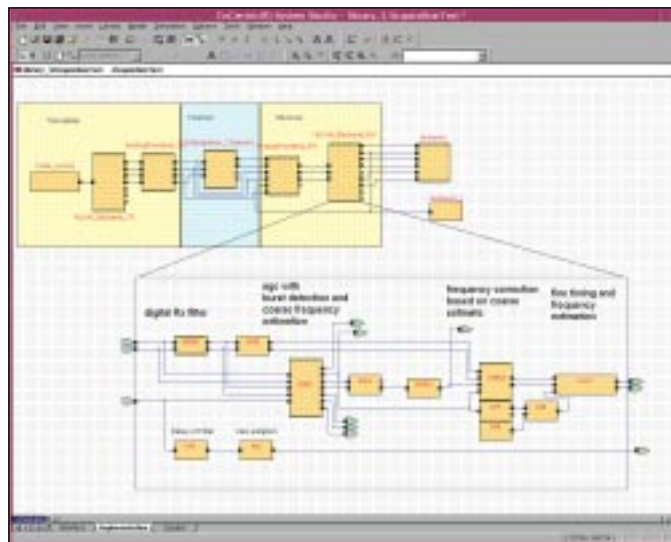


Figure 2: WLAN Reference Design in System Studio

handled correctly. A typical example for this is a multi-processor system. Typically, the users observe that the bus might still have idle cycles in the range of 10-30, percent, but the overall system performance goes down because the processors cannot access the memory in time, when the bus is taken.

So a static analysis of this system would result in the assumption that with 30 percent idle cycles, the processors still have their full processing power and will not wait for data. In the final system, the processors could be stalled most of the time. Transferring this to the system integration in the lab, you can imagine listening to your MP3 audio and, once in a while, hear the audio become interrupted. "Is there a problem?"

What is Algorithm Closure?

Let's take the wireless algorithms in the residential gateway example and discuss the answers we need to get at the algorithmic level before we can proceed to look into a possible implementation. The input of this step is a paper spec of all possible channel characteristics supported by the supported standards, the frame formats at the input and output of the physical layer (PHY) and the expected bit error rate (BER) performance at the output of the receiver. The target performance is typically a set of values, which characterize the BER over the channel characteristic. The channel characteristic is given by the signal-to-noise ratio (SNR).

The output of this step is an executable specification for the transmitter and receiver algorithms, which meets the specified BER performance. The executable specification contains distinct blocks, typically in the complexity of a matched filter, an interpolator and a decimator. Characteristic value for bit error



Figure 3: Burst_read functioncall using the DesignWare AMBA bus model

rates are 10-6-10-8, which results in the need of at least 100x106 (100x108) input values for a sufficient statistic. Figure 1 shows an example for two BER curves. One only partly fulfills the targeted performance. See the red graph, which is partly above the expected performance curve (purple graph).

Why is it possible that one or multiple bits are not correctly received? There are two problems, which need to be addressed during algorithm development. First, frequency selective fading occurs in addition to other transmission distortions like noise and Doppler shift. Second, the system contains at least two physically distinct clocks, one in the receiver and one in the transmitter, with frequencies that may differ slightly or greatly.

A system model that describes these effects, is key for success. As part of the transmission scheme, the algorithms inside transmitters and receivers perform a tremendous amount of up and down sampling operations because either more than one bit is transmitted per information bit (coding) or the modulation requires filtering. Besides the up- and down-sampling operations, most of the receiver structures have feedback loops, for example, to adjust dynamically to fre-

quency shifts in the system. If those frequency loops are not fast or accurate enough, the error performance of the receiver degrades.

Timing information

Algorithm development is typically associated with pure functional models, without timing information; however, latency in feedback loops has a crucial impact on the performance of the system. How much timing information is needed to explore different algorithms but still get realistic performance values? C function calls do not have enough information about time, and a detailed RTL model simulates too slowly. To be able to address these issues and efficiently build optimized algorithms requires a data flow paradigm. Here, the user does not model timing explicitly, but looks at how many tokens are needed at the input of each block, for example, to interpolate between multiple values and how many tokens are sent to the next block for further processing.

System Studio provides a stream-driven simulation engine that is optimized for high-speed execution of data flow systems—essential to run 1010 samples. Therefore a simulation runs more than 1000 times faster than the detailed RTL model.

Figure 2 shows the block diagram for the 802.11 WLAN standard, modeled in System Studio using the data flow paradigm. The transmitter, the channel and the receiver are modeled as multiple blocks. In the receiver there are multiple blocks for the analog front end and for baseband processing. More details of block diagram of the baseband receiver block are shown at the bottom of the figure, with blocks such as the matched filter, a burst detection block and multi-stage frequency estimators.

Other key elements

Other key elements for algorithm exploration are the set of parameters chosen for exploration. Parameterized channel blocks with configurable SNR, frequencies and frequency shifts are one set of parameters to characterize multiple different channel states, which can be characterized through clock shifts and distortions.

On the other hand, configurable filter coefficients allow to modify the filter characteristics and to explore different algorithm. If the user does not only target floating point algorithms but also looks into fixed-point implementations, the bit width of input and output values as well as for internal variables can have huge impact on the overall performance.

With the System Studio Tcl script you run multiple different iterations with the same executable, but various different configurations. In these models, you can adjust the clock deviation as well as the transmission rate of the channel. In the same way you can adjust filter coefficients and/or the bit width of the filter input port.

```
### variable: Bit rate in Mbit/s, either 54,  
48, 36, 24, 18, 12, 9 or 6  
set BitRate 6
```

```
### variable: C/N ratio on the channel  
set C_by_N 20  
  
### PSDU length in bytes  
set_value PSDU_Length_Bytes 1  
  
### Length of gaps in between bursts  
set_value Tx_gaplength_samples  
[ expr 640*4+20 ]  
  
### Mixer frequency and sampling clock  
are derived from same oscillator  
### but to be able to simulate the influence  
of both separately single  
### parameters for each are introduced  
### Clock deviation between Tx and Rx  
oscillator max 40ppm = 20ppm+20ppm  
### influence on sampling clock  
set_value ClockDeviation_ppm 40  
  
### Total clock deviation between Tx and  
Rx carrier  
### oscillator max 40ppm =20ppm+20ppm  
set_value CarrierDeviation_ppm 40
```

When the configurations are explored and the system meets the targeted performance, you close on the algorithm.

What is Architecture Closure?

In the case of the residential gateway, the target architecture is a platform containing multiple HW, but also multiple SW parts. The goal of the architecture exploration phase is to find the trade off between algorithms mapped to the dedicated HW (exclusive bus and memory access for high data throughput, limited flexibility) and algorithms mapped to SW (shared resources in terms of processing power, bus and memory access and high flexibility as well as access to late changes).

The residential gateway architecture has hard and soft real-time constraints and the target performance is specified in terms of packet delay. This is either specified through quality of service requirements for

specific connections or due to a limited amount of buffers to store incoming data. If the processing power in one component in the architecture is too low, this will result in higher packet loss than tolerable.

The input of the architecture exploration step is the algorithm specification, preferable as an executable specification and a set of real-time constraints. The output is a detailed macro-architecture with latency constraints for each block, as well as a detailed specification of the communication infrastructure.

Why is the communication infrastructure so important? In the case of the residential gateway, we are not talking about one processor connected through an internal bus with the program and data memory, maybe with an additional DMA controller to allow data access from the outside. Instead, the situation involves multiple processor systems and dedicated co-processor components and multiple DMA controller accessing one or multiple buses.

In multiprocessor systems, at any time during data processing of one processor, one of the other processors with higher priority on the bus can demand the bus. In this situation the processor, which currently uses the bus, has to stop the data transfer and has to wait until it can resume processing. This can have significant impact on the overall system performance. This means in multiprocessor designs, the data communication infrastructure, if not carefully designed, is a potential source for bottlenecks.

RTL is too slow

The fact that problems occur during data processing means that you have to apply about 104-106 test vectors to the system to identify these bottlenecks. Here again RTL is too slow. You need to have at least a

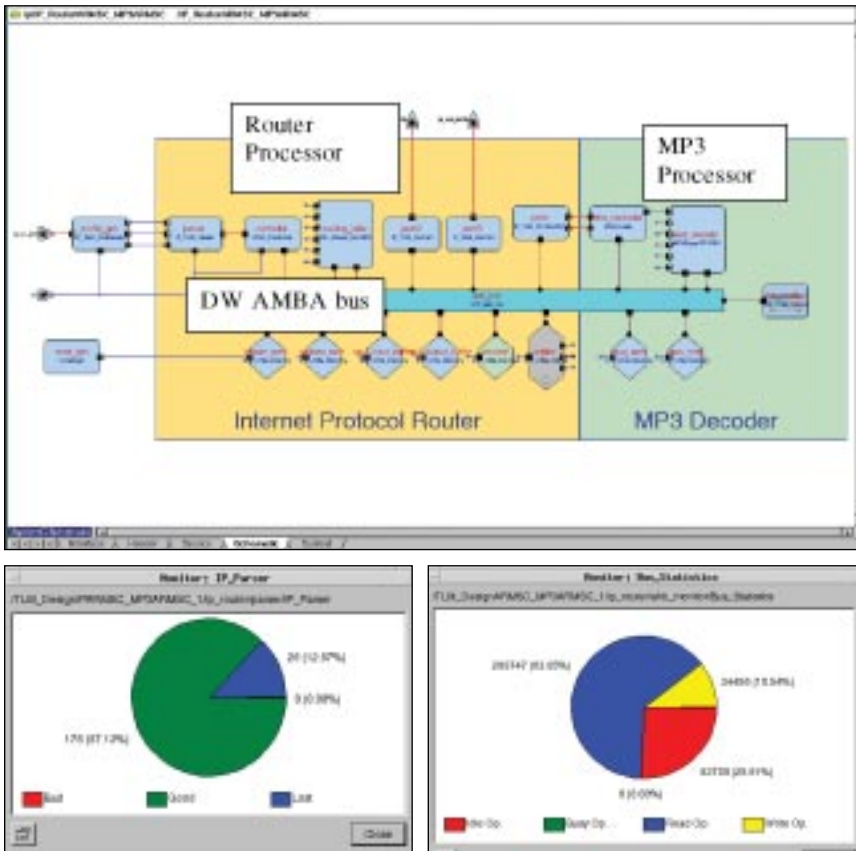


Figure 4: Multiprocessor architecture analyzed in System Studio.

factor of 100 in simulation speed, compared with RTL simulation, to be able to run enough test vectors through the system and get reliable performance values.

How much details are needed to explore different architectures? To identify the communication bottleneck described above, a cycle-accurate model of the bus is essential. Only with a cycle-accurate model of the bus and the processor can the behavior of the SW running on the processor and interacting with the rest of the system be explored.

Identifying bottlenecks

A cycle-accurate bus model handling the data accesses with a single function call (also called transaction, implemented as interface

method call—IMC) gives enough simulation speed and enough details to identify the communication bottleneck. See Figure 4 for a typical multiprocessor platform using Synopsys' DesignWare ARM and AMBA transaction-level models.

Here, you can see the utilization of the bus (right) and the packet loss ratios (left) as a pie chart. The packet loss ratio is too high, even with 30 percent idle cycles. Looking into the transaction trace in Figure 5, the MP3 processor (trans_3) gets preempted before the transmission is finished, because it has a lower priority than the IP router processor (trans_1).

The MP3 processor gets stalled during data access and cannot finish data processing in time. This means

that its memory fills up, data from the IP router cannot be stored and at some point in time the IP router needs to drop packets. Possible solutions are more memory for the MP3 processor, higher priority for its data access or a smaller burst size of the MP3 data accesses. The options need to be explored.

Beside the priorities and the memory sizes for the header and payload memory, #wait cycles per read and write access for the memory, bus bandwidth or burst size for the MP3 stream are important parameters in this system. With these parameters you can run multiple simulations with the same executable, but different configurations for a full architecture exploration. The code fragment in Figure 3 shows a function call to the AMBA bus.

The function call is implemented in the AMBA bus model in a cycle-accurate data access. The bus handles the address decoding and the arbitration, when multiple masters accessing the bus at the same time. The bus will be occupied for the same amount of cycles as the final HW implementation. The DesignWare AMBA bus model implements also the 1-stage pipeline, defined in the AMBA specification. When the configurations are explored and the architecture meets the targeted performance you close on the architecture.

Never without again!

With Synopsys' System Studio supporting C/C++/SystemC modeling and extremely fast simulation, you and your system design team are equipped to attack very complex executable specifications in a short amount of time. Synopsys' Reference Design Kits (RDK) for algorithm optimization for standards such as 802.11 help to quickly develop according to a particular communication standard. Critical

Multistandard Design

as well are accurate architecture models from silicon vendors. Using early versions of TI's SystemC OMAP models we demonstrated to users of System Studio how architecture design based on OMAP can be done.

The answer of project teams being asked to design complex products without completing the executable system level specification should always be: "Never without again!" ♦

Andrea Kröll is a corporate applications engineer in the Verification Technology Group at Synopsys, Inc., Mountain View, CA. She has built up experience in high-level system architecture modeling throughout her career. This

includes work for Create-it, Germany, in the graphic processors space, as well as work in the area of networking. She built

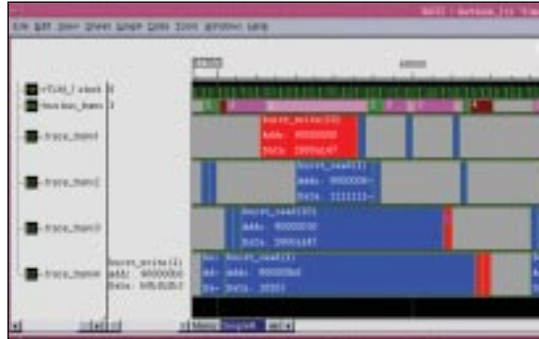


Figure 5: Transaction Trace

a 1st time right networking chip starting with an executable specification at the system level and leveraged this specifica-

tion down to silicon. She holds an M.S. and a Ph.D. in Electrical Engineering from Aachen University of Technology, Germany.

Johannes Stahl is a director of marketing in the Verification Technology Group at Synopsys, Inc., Mountain View, CA. Throughout his career at Synopsys he has specialized in system level design holding management positions for consulting and tool application. His design background is taking system level specifications all the way down to implementation for applications such as cellular phones and video/audio broadcasting. He holds an M.S. and a Ph.D. in Electrical Engineering from Aachen University of Technology, Germany.



DSP Boards and Emulators

...from the leader in TI DSP Development Tools

- 0.5 – 5.0 V
- USB 2.0 at 480 Mb/sec
- Faster load time through QuickLOAD™ technology



ARM 28x 54x 55x 62x 64x 67x OMAP...

FleXDS—High Performance Emulators

- Mobile: PCMCIA, USB2
- Desktop: PCI 510 and 560, EVMs

MSB—Modular Systems and Boards

- Any mix of C6000, Xilinx, AD/DA, I/O
- PCI, cPCI, VME, standalone

TIGER—DSP Development Boards

- C5000 and C6000
- Rich memory and I/O

VIPER—Telecom and VoIP

- Up to 48 cores per board
- PCI, cPCI, H.100/110, MVIP

Video Boards

- Contact us



DSP Research, Inc.
www.dspr.com
information@dspr.com
 Phone: (408) 773-1042

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

| Products | | Applications | |
|------------------|--|---------------------|--|
| Amplifiers | amplifier.ti.com | Audio | www.ti.com/audio |
| Data Converters | dataconverter.ti.com | Automotive | www.ti.com/automotive |
| DSP | dsp.ti.com | Broadband | www.ti.com/broadband |
| Interface | interface.ti.com | Digital Control | www.ti.com/digitalcontrol |
| Logic | logic.ti.com | Military | www.ti.com/military |
| Power Mgmt | power.ti.com | Optical Networking | www.ti.com/opticalnetwork |
| Microcontrollers | microcontroller.ti.com | Security | www.ti.com/security |
| | | Telephony | www.ti.com/telephony |
| | | Video & Imaging | www.ti.com/video |
| | | Wireless | www.ti.com/wireless |

Mailing Address: Texas Instruments
Post Office Box 655303 Dallas, Texas 75265

Copyright © 2003, Texas Instruments Incorporated