Hindawi Publishing Corporation
EURASIP Journal on Embedded Systems
Volume 2008, Article ID 171358, 9 pages
doi:10.1155/2008/171358

Research Article

Novel Methodology for Functional Modeling and Simulation of Wireless Embedded Systems

Emma Sosa Morales, Giorgia Zucchelli, Martin Barnasconi, and Nitasha Jugessur

NXP Semiconductors, Corporate Innovation and Technology, High Tech Campus 37, 5656AE Eindhoven, The Netherlands

Correspondence should be addressed to Emma Sosa Morales, emma.sosa.morales@nxp.com

Received 12 October 2007; Accepted 8 April 2008

Recommended by Christoph Grimm

A novel methodology is presented for the modeling and the simulation of wireless embedded systems. Tight interaction between the analog and the digital functionality makes the design and verification of such systems a real challenge. The applied methodology brings together the functional models of the baseband algorithms written in C language with the circuit descriptions at behavioral level in Verilog or Verilog-AMS for the system simulations in a single kernel environment. The physical layer of an ultrawideband system has been successfully modeled and simulated. The results confirm that this methodology provides a standardized framework in order to efficiently and accurately simulate complex mixed signal applications for embedded systems.

1. INTRODUCTION

Today's telecommunication systems embed high performance analog, mixed-signal, radio frequency (RF), and digital circuitry in a single chip. An example of such an embedded system is the NXP semiconductors wireless USB end-to-end silicon solution based on the ultrawideband (UWB) technology [1].

UWB systems operate at radio frequency in the GHz-range and use advanced modulation methods such as orthogonal frequency-division multiplexing (OFDM), fast frequency hopping techniques, smart radio control, and calibration to maximize link reliability and channel efficiency but also to minimize interference with other services. Growing complexity in the RF front end as well as very accurate digital signal processing is observed. However, the analog and digital subsystems are often developed separately. The combination and tight interaction of RF implementation with the baseband algorithms, required by calibration and control loops, make the design and simulation of such systems a real challenge.

The goal of this paper is to present a methodology for the modeling and the simulation of the physical layer (PHY) of a wireless system within a single simulation environment. This framework enables the validation of the complete physical layer, taking into account the interaction between the analog and the RF models with the baseband algorithms described in the C language.

This paper is organized in sections. In Section 2, the challenges of next generation wireless systems are presented, introducing the UWB as a relevant example. In Section 3, the work related to the applied methodology is summarized. In Section 4, the methodology is detailed. In Section 5, an overview of the behavioral models for the UWB PHY is given. In Section 6, the use of the Verilog Procedural Interface (VPI) as part of the methodology is explained. In Section 7, the results are presented for the UWB PHY followed by the conclusions.

2. PROBLEM DESCRIPTION

In this section, the challenges of today's wireless systems are described and divided into two categories: design challenges, related to the technology and implementation of such systems, and the subsequent modeling and simulation challenges. The requirements that have to be fulfilled by a design and verification methodology are mainly dictated by the simulation challenges of these wireless embedded systems.

2.1. Design challenges

Next generation wireless systems are difficult to design and verify due to extremely demanding requirements such as high transmission frequencies, large bandwidth, sophisticated modulation and coding techniques for high data rate, fast frequency hopping, and low or strictly controlled transmission power. All these features are present in the UWB communication system.

WiMedia proposes the technical specifications for the media access control and the physical layer of the UWB system [2]. A multiband orthogonal frequency division modulation (MB-OFDM) scheme is specified for the physical layer. The information is transmitted in packets which are composed of a variable number of OFDM symbols according to the achievable data rate. A total of 110 subcarriers is used to transmit one OFDM symbol with a channel bandwidth of 528 MHz.

The available spectrum is divided into 14 bands which cover the frequency range from 3.1 GHz to 10.6 GHz. To allow transmissions covering such a large bandwidth, regulation bodies for different geographical regions have put in place severe broadcast restrictions for power spectral emissions of spurious and other interferences. By doing so, UWB devices can make use of an extremely wide frequency band while not emitting enough energy to be noticed by narrowband devices nearby. Due to the restrictions regarding power spectral emissions, the transmitted spread spectrum signal has similar characteristics as wideband noise. Therefore, sophisticated signal processing algorithms and receivers with high sensitivity are required to recover the information especially in the presence of interferers. Furthermore, to increase the robustness, the UWB communication system adopts fast frequency hopping across different frequency bands.

2.2. Modeling and simulation challenges

The design challenges become true modeling and simulation challenges when the physical layer has to be validated. The high frequencies impose very small time steps for the transient simulation which lead to long simulation times. The approach of equivalent baseband modeling [3] does not help to reduce the simulation time due to the large bandwidth of the signals. The complex signal processing and large channel bandwidth increase to a large extent the amount of data that has to be handled by the simulator. In-band and out-of-band interferences and the contribution of noise also add to the complexity of the simulation.

The conventional approach for the design of complex systems consists in dividing it into subsystems, each with limited complexity. The subsystem specifications are defined during the system exploration phase and are then passed to different implementation teams. The design of these subsystems is done by teams of specialists, often distributed all over the world, using multiple and dedicated tools with the consequence that the overall performance of the system is not always monitored. Overspecification often results from adopting a conservative approach to design the different subsystems. Correction loops across multiple domains make the system partitioning critical, requiring a strict definition of the interfaces. Moreover, it is extremely difficult to prove before the tape-out that the designed subsystems fulfill the given requirements in realistic conditions (e.g., including interferers and noise).

Another difficulty encountered when modeling and simulating the whole system is due to the use of multiple languages and simulation environments. Apart from providing a single simulation environment, a system design and verification methodology should also fulfill the following requirements.

(i) Single kernel solution: synchronization problems due to the interaction between different simulation kernels are avoided, resulting in better performance and easier debugging.

(ii) Library independent: models described in standardized languages can be used, thereby reducing the dependency on vendors' libraries. The user has the freedom to decide the level of detail implemented in the model. Refinements to include nonideal effects during both top-down design and bottom-up verification are possible.

(iii) Tool independent: the implementation of the methodology should be compatible with existing mixed signal and mixed abstraction level design environments and flows offered by commercial tool vendors.

(iv) Reasonably fast: tradeoffs between modeled impairments, accuracy, and simulation speed should be possible in the simulation framework for both RF and baseband models.

(v) Aligned with the design community: existing design methodologies, modeling approaches, and standardized languages used by the system and circuit design community should be supported.

(vi) Support vector processing: to handle OFDM and other complex modulation techniques, vector processing should be facilitated by the simulation framework.

The next section gives an overview of the different solutions available to address the simulation challenges described.

3. RELATED WORK

The complexity of today's telecommunication systems triggers electronic design automation tool vendors and universities to provide solutions or study alternatives, ranging from supporting multiple description languages, combining dedicated solvers, or even linking different environments together.

In the past years, significant effort has been spent to develop mixed signal tools combining digital and analog solvers in a single framework and to support mixed signal languages [4, 5]. Nowadays, most major tool vendors provide a solution for the circuit implementation, modeling, and verification of mixed signal applications [6–8].

Most recently, the main effort to tackle the system complexity has been dedicated to combining different levels of design abstraction, facilitating both top-down design and bottom-up verification [9–11]. MATLAB and the C language are often used for the definition of the system specifications and are widely recognized as the languages for system level design [12]. The lack of connection between the dedicated tools used for system level design and the ones used for circuit level implementation has lead to the trend of cosimulation support. Most commercial system level tools have an interface to C language and MATLAB [13–15] and provide cosimulation solutions with mixed signal integrated circuit (IC) design simulators. The list of available links among different tools and environments can easily get lengthy including all possible flavors of analog, digital, and RF solvers. Some examples include MathWorks Simulink and Cadence AMS Designer or Mentor Graphics AdvanceMS, Agilent ADS-Ptolemy and Cadence AMS Designer or Agilent Circuit Envelope, Agilent ADS-Ptolemy and Cadence NcSim or Mentor Graphics ModelSim, CoWare SPD and Cadence AMS Designer.

However, the configuration of the environment and the test bench for the cosimulation is often not trivial and requires extra effort from the architect or the designer. Moreover, converter blocks are often necessary to enable the communication between the simulators [16]. These converter blocks make the interface less transparent. The tight communication between different solvers needed in a cosimulation approach is a potential source of synchronization problems, deteriorating the simulation performances in terms of speed and increasing the difficulty of the debugging process.

A single kernel solution linking system level with mixed signal IC design presents advantages over cosimulation solutions [3, 17]. The possibility of using standardized programming interfaces to call C code functions within hardware description languages (HDL) models overcomes the limitations of cosimulation and guarantees a transparent definition of the simulation set up. To the authors knowledge, the standardized interfaces [18, 19] have since long been used to set up cosimulation with third-party tools [10] and for testing purposes [20, 21]. The innovative approach described in the next section proposes the use of such interfaces to embed digital signal processing in a mixed signal environment without the penalties of cosimulation.

4. APPLIED METHODOLOGY

In this section, the methodology to address the modeling and simulation challenges is introduced.

The methodology is based on the following three elements:

(1) the use of a mixed signal simulator to combine analog and digital signals in a single kernel environment;

(2) the use of behavioral models to describe the functionality of the RF and mixed signal blocks with standardized languages and the use of high-level languages (C/C++) for algorithms;

(3) the use of an environment that allows functional description in combination with circuit level implementation to enable mixed abstraction level simulations.

The mixed signal simulator provides a framework where analog and digital seamlessly come together. Such a framework is obviously useful for the simulation of blocks described at implementation level as transistor netlists and RTL. Most behavioral languages are supported by mixed signal simulators, allowing both top-down and bottom-up design methodologies: systems can be modeled and simulated at different levels of design abstraction.

Through standardized languages, dedicated behavioral models can be developed extending the commercially available libraries. Since the simulation speed depends on the amount of details captured (e.g., impairments), models can be refined to achieve the desired compromise between accuracy and simulation speed. In-house developed models are intellectual property (IP), representing a valuable asset that can be exchanged among design groups, thereby increasing the level of expertise and IP reuse. Examples of behavioral models for the functional description of the UWB RF transceiver are presented in Section 5.

In a mixed signal IC design environment, Verilog and Verilog-AMS are often used to model the digital and analog implementation. However, for more complex systems such as OFDM modems, where many computations are done using vector or matrix operations, these HDL languages do not offer sufficient semantics for efficient functional modeling [3, 17]. Languages such as C or C++ are often used for this purpose. The Verilog Procedural Interface (VPI) [18] allows calling algorithm descriptions in C language from within Verilog or Verilog-AMS modules. In Section 6, the interaction between a given C function and a Verilog or Verilog-AMS module through the VPI is described.

Since the VPI is part of the IEEE 1364 standard for Verilog HDL and supported by most tools vendors, the applied methodology is tool-independent and can be implemented in several commercially available simulation environments.

Figure 1 shows the philosophy behind the methodology. Within a single environment, C language, behavioral models, transistor netlists, and RTL can be brought together. This methodology offers a framework in which the challenges of next generation communication systems can be handled. The standardized VPI for mixed abstraction level simulations combined with the standardized HDLs for behavioral modeling and a mixed signal environment results in a differentiating methodology compared to other solutions available on the market.

5. MODELING THE UWB PHYSICAL LAYER

In this section, the models for each block of the UWB PHY and their interfaces are described. The baseband algorithms, compliant with the WiMedia specification, are first explained. The functional description of each constituent of the RF transceiver is then summarized. These models have simple characteristics and introduce only fundamental

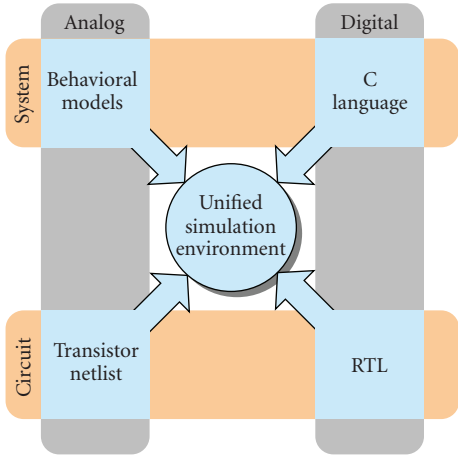

Figure 1: With the applied methodology, different levels of design abstraction for the analog and digital domains come together in the same framework.

impairments. Within the framework, further refinement of the models is possible.

Figure 2 shows the block diagram of the UWB PHY. Behavioral models have been developed in Verilog-A or Verilog-AMS for the RF transceiver and in C language for the baseband. These models are combined together in a single test bench for system validation. Within the same test bench, the functional models can be replaced by transistor level descriptions for circuit-level validation in the system context. This framework therefore facilitates complex mixed signal and mixed abstraction level simulations.

5.1. Baseband models

The *baseband transmitter* generates the OFDM signal compliant to the physical layer specification of the WiMedia proposal. This requires algorithms for digital signal processing: scrambling, forward error correction, code interleaving, constellation mapping, OFDM modulation using inverse fast Fourier transformations, insertion of pilots, guard carriers addition, and cyclic prefix addition.

This functional model is described as a C code function that generates two vectors representing one frame of the in-phase I and quadrature Q components of the OFDM signal. It also generates the signal which controls the local oscillator frequency of the *IQ Modulator*. The transmission rate and the length of the packet are among the parameters of the baseband transmitter model.

The *baseband receiver* implements the inverse operations of the baseband transmitter. In order to recover the transmitted data, the following procedures are used: timing and frequency offset correction, OFDM demodulation using fast Fourier transformations, channel correction, constellation demapping (digital demodulation), deinterleaving, forward error correction decoding and descrambling.

The functional model of the baseband receiver is implemented as a C code function. It takes as inputs the I and Q components of the OFDM signal and the information passed from the synchronizer in order to extract the payload from the packet. The model has the same parameters as the baseband transmitter.

The *synchronizer* calculates, per sample of the I and Q components, the information needed by the baseband receiver to identify the payload in the packet. The synchronizer also recovers the hopping sequence and controls the local oscillator frequency of the *IQ demodulator*.

The functional model of the synchronizer is implemented in C language and does not have any user-defined parameters.

5.2. RF transceiver models

The *IQ modulator* and *demodulator*, respectively, up- and down-convert the I and Q samples with a carrier frequency that changes from symbol to symbol. A defined hopping sequence is used to generate the carriers that drive the mixer for the frequency conversion.

The Verilog-AMS models of the modulators [22] include nonidealities such as third-order distortion and gain compression. To simplify the model interface, the local oscillator is described as an internal signal. The parameters of the model are the input impedance, output impedance, voltage conversion gain, and input referred third-order intercept point. The demodulator model also contains a low pass filter to reject the signal image of the demodulated signal at the higher frequency.

The *TX filter* and the *RX filter*, respectively, filter the symbols generated by the baseband transmitter and recovered by the demodulator.

The Verilog-AMS models of the filters implement a linear Laplace function with poles and zeros as arguments. The poles and zeros are extracted from the circuit implementation after a pole-zero analysis. Input and output impedances are specified as parameters.

The *PA* and *LNA* amplify the signal power to compensate for the channel attenuation.

The Verilog-A model of the amplifiers includes second-order distortion, third-order distortion, and gain compression. The parameters of the models are the input impedance, output impedance, voltage conversion gain, input referred third-order intercept point, and input referred second-order intercept point.

The *channel* represents the pathway over which data is transferred from the antenna of the transmitter to the one of the receiver.

A Verilog-A model describes an ideal channel with the attenuation as a parameter. Interference, fading, multipath delay spread, and other nonidealities can be included in the model.

5.3. Interfaces

The signal types at the interfaces of the Verilog-AMS models are either *wreal* or *electrical*. Signals of type *wreal* are discrete in the time domain and are handled by the discrete event solver. *Wreal* is therefore preferred at the baseband interface. However, the *electrical* type is more appropriate for the

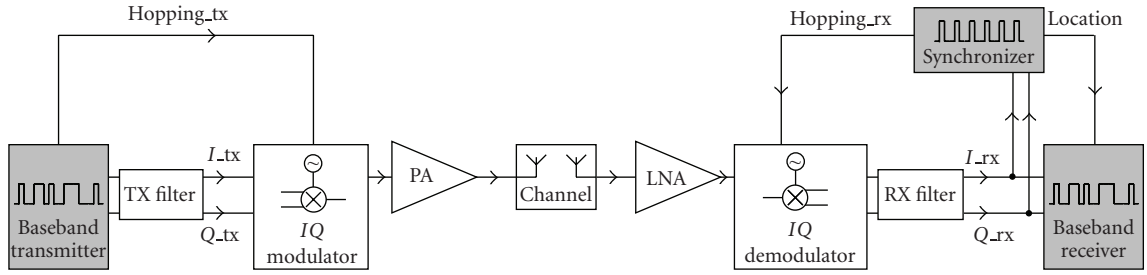

Figure 2: Block diagram of the UWB PHY with the baseband (in light grey) and the RF transceiver.

```
void vpi_wrapper ()
{
vpiHandle      taskHandle, argIter;
vpiHandle      inHandle, outHandle;
s_vpi_value    inVal, outVal;
double         *buffer;

taskHandle = vpi_handle(vpiSysTfCall, NULL);
argIter = vpi_iterate(vpiArgument, taskHandle);
if (argIter) {
   inHandle = vpi_scan(argIter);
   outHandle = vpi_scan(argIter);
   if (inHandle && outHandle ) {
      inVal.format = vpiIntVal;
vpi_get_value(inHandle, &inVal);
      /* Call to user-defined C function */
      user_func(inVal.value.integer, buffer);
      outVal.format = vpiRealVal;
      outVal.value.real = *buffer;
      vpi_put_value(outHandle, &outVal, NULL,
                vpiNoDelay);
      ...
}
```

Example 1

description of analog and RF signals. This type is handled by the analog solver (continuous time) which has to solve the differential equations from Kirchoff's laws at all electrical nodes.

6. EMBEDDING C CODE

This section describes the embedding of C code in a mixed signal IC design environment. The interaction between a C code function and a Verilog or Verilog-AMS model is through the Verilog procedural interface. The main features of the VPI are described in Section 6.1. The introduction of the notion of time within the VPI and the handling of the event synchronization and the vector processing for the UWB PHY are detailed in Section 6.2.

6.1. Verilog Procedural Interface

The Verilog Procedural Interface is the programming interface for the Verilog hardware description language and standardized under IEEE 1364. The VPI allows the writing of applications to create simulator tasks or functions that can be called from within Verilog HDL designs. VPI applications are dynamically loaded by the simulator, thereby making the system task or function part of the simulator executable. These user-defined system tasks are called from the HDL design during simulation.

In the presented methodology, the VPI is used to embed C code functions describing the baseband algorithms in an IC design environment. All the functionalities and the flexibility of the C language with respect to dynamic vector allocation (pointers) are supported within the VPI making it suitable for the support of vector-based digital signal processing.

Figure 3 shows the interaction between a C code function and a Verilog or Verilog-AMS module through the VPI. The functionality to be imported is described as a C code function (inner kernel in Figure 3). By using the access and utility routines provided by the VPI, a wrapper is built around this C code function. The wrapper handles the inputs/outputs and allows the C code function to interact with the ports and variables of the Verilog-AMS module. Two functions are used for this purpose: *vpi_get_value()* and *vpi_put_value()*. *vpi_get_value()* is used to read a value from the Verilog-AMS module and to make it available to the C code function. *vpi_put_value()* allows the writing of the results calculated by the C code function to the Verilog-AMS module. Within the wrapper, any necessary signal type conversion is handled. No additional models is necessary to translate the signals as compared to cosimulation solutions where dedicated interfaces (e.g., connect modules) are often required.

In Example 1, the code in C language shows an example of how to combine the utility functions provided by the VPI in order to build the wrapper.

Once the wrapper is built, it needs to be registered as a simulator task or function and compiled into a shared object library. In our case, *gcc* has been used for compilation and linking and provides all standards options for debugging the C code function. The debugging possibilities of the VPI functions within the IC design environment are, however,

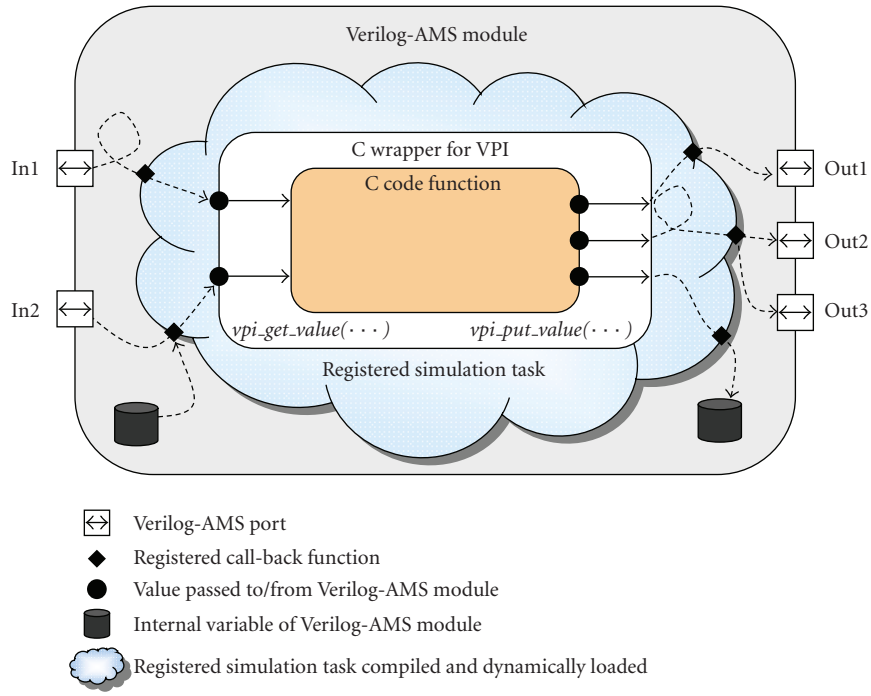

Figure 3: Schematic representation of the interaction between the Verilog-AMS module and the C code function wrapped using VPI routines.

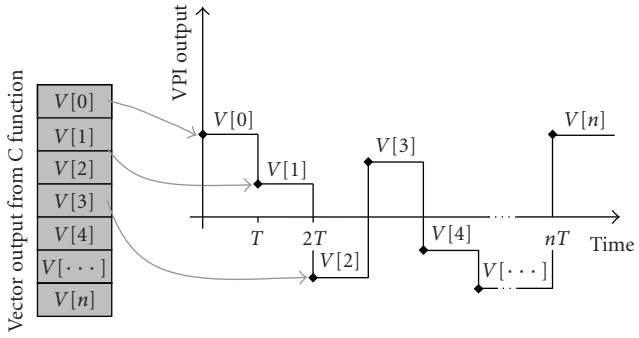

Figure 4: Process of unrolling the vector generated by the C code function in the time domain.

tool-dependent. The simulator task or function can be called from a Verilog or a Verilog-AMS module. The shared object is dynamically loaded during the elaboration phase.

In order to embed C code in a mixed signal IC design environment, the user should have basic knowledge of the C language as well as hardware description languages such as Verilog or Verilog-AMS. Familiarity with C development environments as well as with mixed signal simulators could also prove to be useful. The steps of creating the wrapper, registering the simulator task, compiling, and linking into a shared object could be fully automated by tool vendors in order to embed a C code function in a mixed signal IC design environment.

Table 1: Error on sampling time and characteristic frequency as a function of precision.

Precision	T (ns)	ΔT (ps)	Δf (MHz)
1 ns	2	106.060606⋯	29.568
1 ps	1.894	0.060606⋯	0.017

6.2. Time management

With the described methodology, the wrapper is used for introducing the concept of time that is otherwise not known in the C code function. For a simulation in the time domain, each sample is associated with a time stamp and is handled as an event by the digital solver. The events are detected with a finite accuracy. According to the chosen precision, a finite number of digits are used to represent the time. The timing error introduced with this approximated representation can be interpreted as a fixed jitter on the sampling time. In long simulation runs, this error can become relevant due to its cumulative behavior. Synchronization problems arise when events on a signal are missed due to the signal being generated and sampled with clocks having the same sampling rate but specified with different precisions.

The simulation time and the results depend on the chosen precision. For the UWB system, the sampling time of the baseband signals is 1.8939393⋯ nanoseconds (1/528 MHz), rounded to T by the digital solver. The resulting error ΔT in the time domain affects the signal representation in the frequency domain [23]. Table 1 shows the relation between

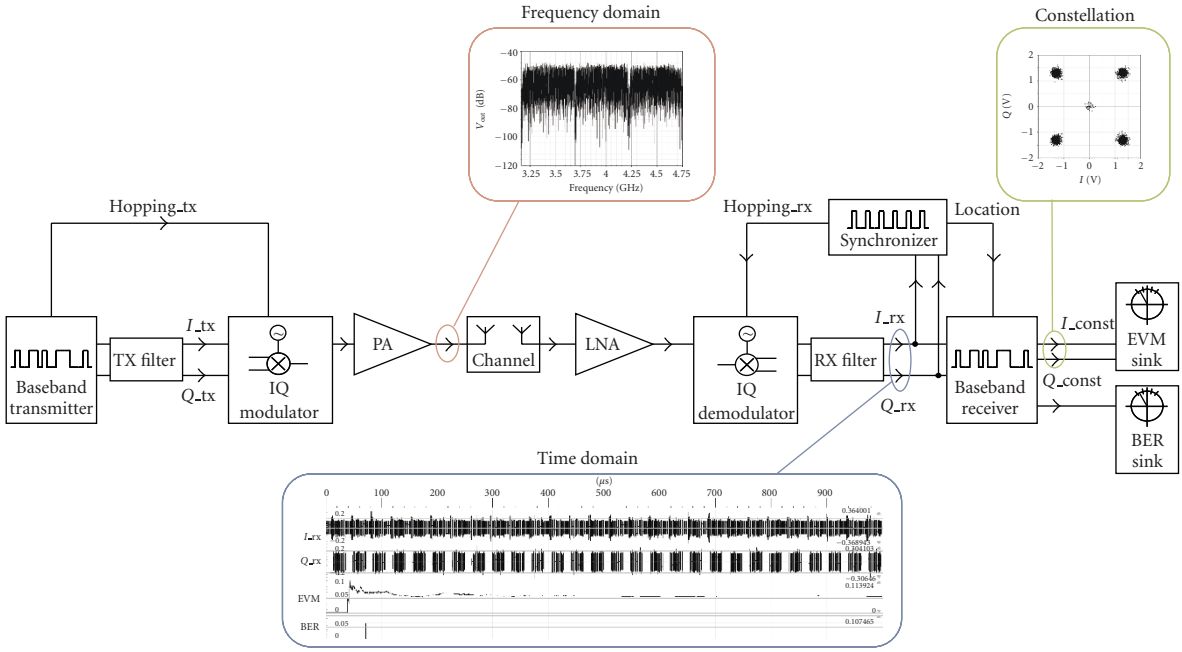

Figure 5: Abstracted view of the test bench and simulation results.

the chosen precision, the error introduced on the sampling time, and the consequent error Δf on the characteristic frequency of the signal.

For a precision of 1 picosecond, the frequency error relative to the signal bandwidth is 17 kHz over 528 MHz which is negligible. Furthermore, since the methodology makes use of only one solver, the same precision is applied to all signals in the system, thereby avoiding any synchronization problem.

The C code functions that describe the baseband algorithms process vectors representing the signal packet or frame without any notion of time. However, the capabilities of vector processing from within Verilog modules are limited with the consequence that the vectors have to be processed as a sequence of samples in the time domain. The unrolling and the packing of the vectors are preferably done directly in the wrapper using dedicated functions for controlling events and adding time information.

The vectors generated by the C code function of the baseband transmitter are unrolled in the wrapper to a sequence of samples transmitted with a specific time stamp as shown in Figure 4. The wrapper of the baseband receiver samples the signals and packets them into a vector to be processed by the C code function. This approach is very flexible since the timing information as well as the length of the vector can be adapted runtime according to the system parameters and are not statically defined as in the case of synchronous data flow solvers [24].

The data generated by the baseband transmitter are fed through the RF chain and are then sampled and recollected by the baseband receiver. For the correct functioning of the system, the sampling frequency of the signal is known and is the same for both the baseband transmitter and receiver.

Due to the fact that a discrete event solver is used for digital signal processing, the sampling rate is not a property of each signal as it is for example in the case of a synchronous data flow solver, where the rate is part of the scheduling process. The knowledge of the correct sampling time is essential for all signals that must be sampled by the successive blocks in the chain. The sampling time could have been retrieved with a specific function to recover the clock on the receiver side. In our case, to limit the complexity of the simulation, a clock is generated in the baseband transmitter and triggers the sampling of the signals in the successive blocks of the chain.

7. SIMULATION RESULTS

The test bench for the functional simulation of the UWB PHY and the performance results are described in this section.

The methodology is implemented using AMS Designer [7], the mixed signal simulator of Cadence.

Figure 5 shows an abstracted view of the test bench with the simulation results. The test bench combines the functional models of the baseband transmitter and receiver in C language, of the RF transceiver in Verilog-A or Verilog-AMS together with the performance metrics. The frequency spectrum is calculated using discrete fast Fourier transformations applied to the result of the transient analysis. The constellation diagram is drawn by plotting the quadrature output against the in-phase output of the baseband receiver. The presented simulation results are qualitative and the parameters of the functional models can be dimensioned

according to the functional requirement specifications of the UWB PHY.

The performance of the UWB system is evaluated in terms of metrics such as bit error rate (BER) and error vector magnitude (EVM). To evaluate the simulation performance of the complete UWB PHY, a transient analysis is run for 5 milliseconds which requires a simulation time of 3 hours. At the chosen rate of 110 Mbps, this corresponds to approximately 143 packets of data and 590304 payload bits. The number of processed bits is enough to evaluate the EVM. However, the accurate estimation of the BER requires a longer transient analysis.

To estimate the overhead introduced by the VPI, another transient analysis of 5 milliseconds is done using a test bench consisting of only the baseband models in C language. The total simulation time required by this setup is less than 3 minutes. From the difference between the two run times, it can be deduced that the contribution of the VPI to the simulation time is negligible.

The framework shows the possibility of implementing feedback loops. In the test bench, the synchronizer recovers the hopping sequence from the received data and controls the frequency of the local oscillator in the demodulator, showing the tight interaction between the baseband and the RF chain. This feedback loop is implemented without introducing artificial delays in the system description.

The analog part of the UWB system is simulated with a maximum time step around 25 picoseconds which corresponds to a simulation bandwidth of 20 GHz. Such a large simulation bandwidth not only allows the capture of the spectral regrowth but also facilitates the addition of interferers for coexistence studies.

8. CONCLUSIONS

The evolution of wireless communication standards has lead to systems in which analog, RF, and mixed signal functionalities have to be combined with digital signal processing algorithms, calibration, and correction loops. The functional simulation of such systems is a real challenge when using existing methodologies and simulation tools that have limited capabilities in handling mixed signal and mixed abstraction level designs.

This paper has presented a methodology that enables functional system verification in the time domain. The use of a mixed signal simulator and the Verilog procedural interface brings together analog and RF behavior in Verilog-AMS models with baseband algorithms in C language, enabling a true mixed signal and mixed abstraction level simulation environment. A single kernel framework is used avoiding any synchronization problem. The use of standardized languages and interface makes the methodology library and tool independent.

A demanding system, the ultrawideband physical layer, has been selected to evaluate the methodology. Analog behavioral models of the RF transceiver together with functional models of the baseband allow raising the level of design abstraction and coping with the system complexity. The test bench serves as a framework for complex mixed abstraction level simulations, where blocks at transistor level can be verified in the system context. Algorithm exploration for the calibration of impairments and compensation techniques in the RF transceiver is also possible using the same test bench. Feedback loops are simulated without any artificial delay introduced by the simulation framework.

The results shown in this paper confirm that the methodology can handle the complexity of modern multidisciplinary systems. A simulation platform is available from which the system architects, the designers of integrated circuits, and the algorithms can benefit, helping them to manage the complexity of next-generation wireless embedded systems.

REFERENCES

[1] NXP Semiconductors, UWB product literature, http://www.nxp.com/acrobat_download/literature/9397/75015695.pdf.

[2] "High Rate Ultra Wideband PHY and MAC standards," standard ECMA 368.

[3] S. Joeres and S. Heinen, "Mixed-mode and mixed-domain modelling and verification of radio frequency subsystems for SoC-applications," in *Proceedings of the IEEE International Behavioral Modeling and Simulation Workshop (BMAS '05)*, pp. 54–59, San Jose, Calif, USA, September 2005.

[4] Verilog-AMS Language Reference Manual. Analog & Mixed-Signal Extensions to Verilog HDL. Version 2.1, Accellera, http://www.accellera.org/.

[5] IEEE Standard VHDL Language Reference Manual IEEE Std 1076, 2000.

[6] Mentor Graphics, product data sheet, http://www.mentor.com/products/ic_nanometer_design/ms_circuit_simulation/advance_ms/upload/ADMS_Datasheet.pdf.

[7] Cadence Design Systems, product data sheet, http://www.cadence.com/datasheets/virtuoso_mmsim.pdf#page=7.

[8] Synopsys, product data sheet, http://www.synopsys.com/products/discoveryams/discoveryams_ds.pdf.

[9] U. Knochel, T. Markwirth, A. Hartung, R. Kakerow, and R. Atukula, "Verification of the RF subsystem within wireless LAN system level simulation," in *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE '03)*, pp. 286–291, Munich, Germany, March 2003.

[10] S. Mu and M. Laisne, "Mixed-signal modeling using Simulink based-C," in *Proceedings of the IEEE International Behavioral Modeling and Simulation Workshop (BMAS '05)*, pp. 128–133, San Jose, Calif, USA, September 2005.

[11] A. Sayinta, G. Canverdi, M. Pauwels, A. Alshawa, and W. Dehaene, "A mixed abstraction level co-simulation case study using systemC for system on chip verification ," in *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE '03)*, pp. 95–100, Munich, Germany, March 2003.

[12] G. Arnout, "C for system level design," in *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE '99)*, pp. 384–386, Munich, Germany, March 1999.

[13] Agilet EEsoft EDA, http://eesof.tm.agilent.com/.

[14] The MathWorks, http://www.mathworks.com/.

[15] CoWare, http://www.coware.com/.

[16] U. Eichler, U. Knöchel, S. Altmann, W. Hartong, and J. Hartung, "Co-simulation of Matlab/Simulink with AMS Designer in System-on Chip Design," SNE16/2 2006.

[17] S. Joeres and S. Heinen, "Functional verification of radio frequency SoCs using mixed-mode and mixed-domain simulations," in *Proceedings of the IEEE International Behavioral Modeling and Simulation Workshop (BMAS '06)*, pp. 144–149, San Jose, Calif, USA, September 2006.

[18] C. Dawson, S. K. Pattanam, and D. Roberts, "The verilog procedural interface for the verilog hardware description language," in *Proceedings of the IEEE International Verilog HDL Conference (IVC '96)*, pp. 17–23, Santa Clara, Calif, USA, February 1996.

[19] F. Martinolle and A. Sherer, "A procedural language interface for VHDL and its typical applications," in *Proceedings of the International Verilog HDL Conference and VHDL International Users Forum (IVC/VIUF '98)*, pp. 32–38, Santa Clara, Calif, USA, March 1998.

[20] L. Chun, Y. Jun, G. Gugang, and S. Longxing, "Domain fault model and coverage metric for SoC verification," in *Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS '05)*, vol. 6, pp. 5662–5665, Kobe, Japan, May 2005.

[21] P. A. Riahi, Z. Navabi, and F. Lombardi, "A VPI-based combinational IP core module-based mixed level serial fault simulation and test generation methodology," in *Proceedings of the 12th Asian Test Symposium (ATS '03)*, pp. 274–277, Xian, China, November 2003.

[22] J. E. Chen, "Modeling RF Systems," http://www.designers-guide.org/.

[23] R. B. Staszewski and R. Staszewski, "VHDL simulation and modeling of an all-digital RF transmitter," in *Proceedings of the 5th International Workshop on System-on-Chip for Real-Time Applications (IWSOC '05)*, pp. 233–238, Banff, Canada, July 2005.

[24] E. A. Lee and D. G. Messerschmitt, "Static scheduling of synchronous data flow programs for digital signal processing," *IEEE Transactions on Computers*, vol. 36, no. 1, pp. 24–35, 1987.