Paper

A seamless software defined radio development flow for waveform and prototype debugging

Ernst Martin Witte, Torsten Kempf, Venkatesh Ramakrishnan,
Gerd Ascheid, Marc Adrat, and Markus Antweiler

Abstract—With the increasing number of wireless communication standards flexibility has gained more and more importance which has lead to the software defined radio (SDR) concept. However, SDR development has to face many challenges, among them are the questions how SDR systems can be designed to achieve flexibility, architectural efficiency, energy efficiency and portability at the same time. These requirements result in very elaborate architectures and a highly increased design complexity. To cope with such complexity, we proposed an SDR development flow. During the development of such SDR, debugging becomes more efficient on a prototype hardware implementation than on a simulation model. However, error analysis on a prototype suffers from strong limitations like a reduced state visibility. In this paper, an extension to the SDR development flow is presented and successfully applied to an example SDR. It allows for an efficient error analysis with the SDR simulation model by the feedback of stimulus data from the prototype.

Keywords— software defined radio, prototype platform, waveform development environment, electronic system level simulation, waveform debugging, stimulus feedback.

1. Introduction

The demand for software defined radio (SDR) systems originates from the fact that today's radio communication systems are designed for a single (or at least a very small number of) waveform specification(s) only. This causes severe interoperability issues. In order to achieve *interoperability* the key idea is to add *flexibility* to the radio hardware platform. This allows to run many standards on the same hardware and with this, enables the communication partners to easily agree on a common waveform. The flexibility issue can be solved by adding *programmability* and *reconfigurability* to the hardware platform. However, there are well known trade-offs between flexibility, performance and energy efficiency. Heterogeneous multiprocessor systems on chip (MPSoCs) are a widely accepted candidate to cope with this challenge. Compared to traditional chip designs, this approach results in a highly increased design complexity.

While the goals of *flexibility, programmability,* and *reconfigurability* are major hardware-related challenges in designing an SDR system, an additional more software-related key topic is *portability*. *Portability* means that a waveform implementation can be transferred from one platform to another. *Portability* of waveforms is of particular importance if waveforms shall be exchanged between communication partners with different hardware platforms. For instance, currently efforts have been started within NATO to establish a *waveform library* for the coalition partners.

The software framework to realize these features has been set by the software communications architecture (SCA) [2]. However, especially the hardware and *portability* related problems raise a couple of issues, which are not addressed by the SCA so far and have to be solved by the waveform and system designers. For example, it has not been specified how to generate software (SW) and hardware (HW) code that is applicable to a given specific SDR platform.

In order to solve these problems, a concept for a seamless design flow for a waveform development environment (WDE) starting from a waveform description language (WDL) [3, 4] down to the (semi-) automatic generation of SCA-compliant SW/HW code for implementation on a (more or less) arbitrary SDR platform has been proposed in [1]. Such a concept can be the basis for the *waveform library* mentioned above. It contains waveforms specified in the WDL which can be ported with reasonable efforts by the nations to their national SDR platforms by (semi-) automatic code generation.

The key idea of the concept is an orthogonalization of the waveform description (functionality and topology), the hardware platform design and the mapping to an arbitrary hardware platform. These three tasks lead to an iterative process of implementing, testing and taking design decisions.

Due to its complexity, the SDR design process can benefit from the hardware/software co-design concepts of electronic system level (ESL) [5] simulation platform ("virtual prototype"). It allows designers to iteratively refine the complete SDR MPSoC model on different abstraction levels depending on the intended use. For example, fast instruction accurate models support the software developer while cycle accurate hardware models are needed by the hardware designer. During hardware implementation, additional models will be required, depending on the abstraction level used in the ESL simulation platform. Although the WDE concept has a focus on a consistent design *flow*, the *functional correctness* of a prototype hardware implementation cannot be guaranteed. Therefore, implementation errors will also be detected on the HW prototype and must be analyzed and resolved. However, due to the com-

plexity of SDR systems, debugging via standard debug interfaces such as joint test action group (JTAG) is difficult. The visibility of internal states is limited. A debug halt of one component can break the synchronicity between parts of the system, for example a transmitter continues to run while the receiver is being debugged which results in lost samples at the receiver (real time requirements). Therefore, this approach of debugging the SDR prototype results in very high development and analysis effort.
However, often errors that are only identified in the prototype implementation, still can be analyzed in the ESL simulation model. This makes error analysis significantly faster and more efficient by allowing the complete visibility of internal states, and guarantees the synchronicity of all system parts. In order to enable the system level analysis of errors found in later development stages such as a prototype or hardware implementation, it is necessary to drive the system simulation into the same state as the hardware. This can only be achieved by a suitable stimulus recorded on the prototype and fed back into the system simulation. In this paper, the extension of the WDE concept with stimulus feedback loops as shown in Fig. 1 will be presented.

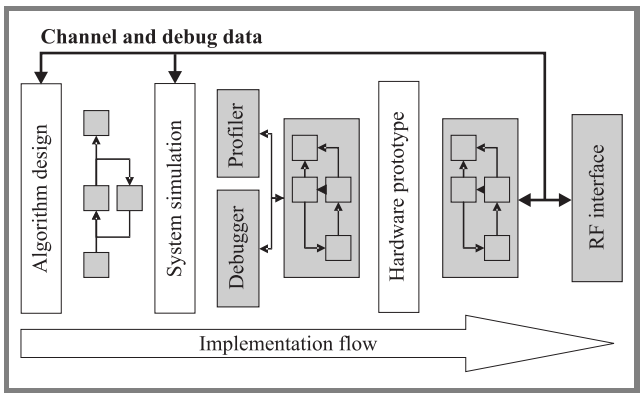


Fig. 1. Prototype toolflow integration by feedback loops.

Section 2 will briefly introduce the concept for a seamless waveform development and highlight development issues arising when realizing a hardware implementation, e.g., on a prototype platform. In Section 3 the development flow of an example SDR prototype system will be presented and analyzed with respect to debugging properties. Based on these investigations, Section 4 will present the realization of the stimulus feedback loops.

2. Concept for seamless waveform development

In [1] a concept for a seamless design environment for SDRs starting from a waveform description down to the implementation onto an SDR hardware platform has been proposed. In contrast to formerly known approaches [3, 4], it is neither limited to a single aspect like waveform descriptions nor to a specific tool.

2.1. Concept description

The concept is based on four key elements as illustrated in Fig. 2.

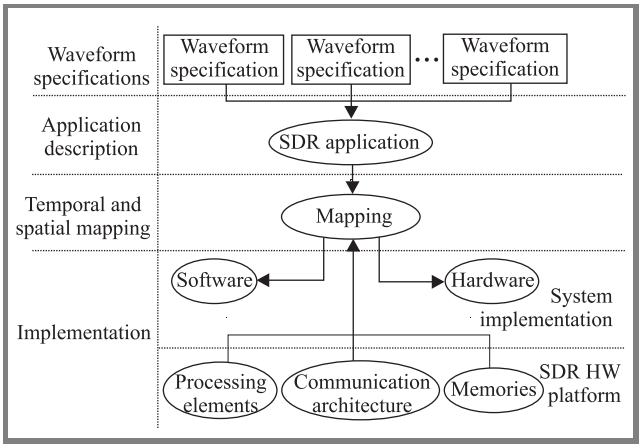


Fig. 2. Design flow from application description to implementation.

Waveform specification. Firstly, a waveform application, typically given as a written document (specification) is implemented based on general programming language structures that are applicable for both software and hardware description. The waveform application is decomposed into functional *blocks* (e.g., voice coding, *forward error correction*, and modulation) and communication *edges*. Every *edge* represents a specific path for data exchange between two *blocks*. The full assembly of *blocks* and *edges* forms a *topology graph*. The *topology graph* summarizes the data dependencies of the functional *blocks* and combines them to a complete waveform application.

The SDR HW platform description. Secondly, the description of an arbitrary SDR hardware platform composed of *processing elements*, *communication architectures* and *memories* is required. The *processing elements* are the devices which perform the signal processing. They can be grouped into *programmable* (e.g., GPP, DSP), *reconfigurable* (e.g., FPGA), and *configurable* (e.g., ASIC) devices. *Communication architectures* describe the data exchange capabilities and are mainly characterized by their *type* (e.g., wires, point-to-point) and their *interfaces*.

Mapping. Thirdly, the mapping of a waveform application onto an arbitrary platform has to be considered in *temporal* (when to execute) and *spatial* (where to execute) manner. The mapping has a major influence on system performance and is a key issue for *portability*. Throughout the mapping process, the functional *blocks* of the waveform decomposition are mapped to the *processing elements* and the *edges* to the *communication architectures*, respectively. It is possible that several *blocks* are mapped to the same *processing element* such that a scheduling of the functional execution becomes necessary. It is also possible that one *block* is split up into subtasks which are processed in parallel by different elements. Criteria for the feasibility analysis of such mappings have been discussed in [6].

Code generation. Last but not least, the fourth step provides the link from the application and platform description towards the system implementation by a (semi-) automatic generation of SW/HW code [1].

2.2. SDR development and prototyping

The proposed concept highlights a seamless design flow from the waveform down to the SDR implementation. Despite of a minimization of design errors by a seamless design flow, such a highly complex SDR prototype will likely suffer from still undetected errors, even if the SDR application and/or hardware platform have been fully verified separately. Such a prototype implementation might reveal issues and corner cases, e.g., related to synchronization, latency problems, etc.

The proposed concept requires the developers to define the SDR application, the SDR hardware platform and the mapping of waveform tasks to processing elements. Therefore, the concept does not eliminate the developer's expertise from these tasks.

However, with these descriptions, the concept enables the generation of the full system simulation and even the hardware configuration (e.g., FPGA bit streams, etc.), if a sufficiently detailed hardware description is provided. Thus, an iterative design space exploration will become highly efficient. The system simulation can provide detailed information about the performance (throughput, latency, bus/CPU load, etc.) of single components or the whole system. With this analysis the developer is able to optimize the mapping, the hardware platform and/or the waveform, depending on his development focus. Furthermore, debugging benefits from the full visibility of internal states. A prototype workbench realizing this concept has been introduced in [7]. The ESL models for the system simulation can be generated for the complete SDR. Fast field programmable gate array (FPGA) prototyping is enabled by the automatic generation of implementation files for the Xilinx embedded development kit (EDK) [8].

The SDR hardware platform can be considered as MPSoC. Designing such an MPSoC is a challenging task, which is addressed separately in research. Recent research activities have put forth the paradigm of ESL [5]. Complete systems (MPSoCs) are assembled and can be analyzed and verified by simulation. To cope with the enormous complexity, simulations are run at a higher level than register transfer level (RTL) using transaction level modeling (TLM).

Figure 3 illustrates the system development flow based on the proposed concept. After finalizing the description of SDR waveform application and hardware platform, developers can map the application to the hardware platform. The generated output is the system implementation, which can either be an ESL simulation model or an FPGA prototype. The mapping must meet design constraints. Initially, estimates will be used which are based on a high level of abstraction of the hardware platform behavior. Then, ESL simulation will be used to verify functional correctness and to check compliance with performance constraints,

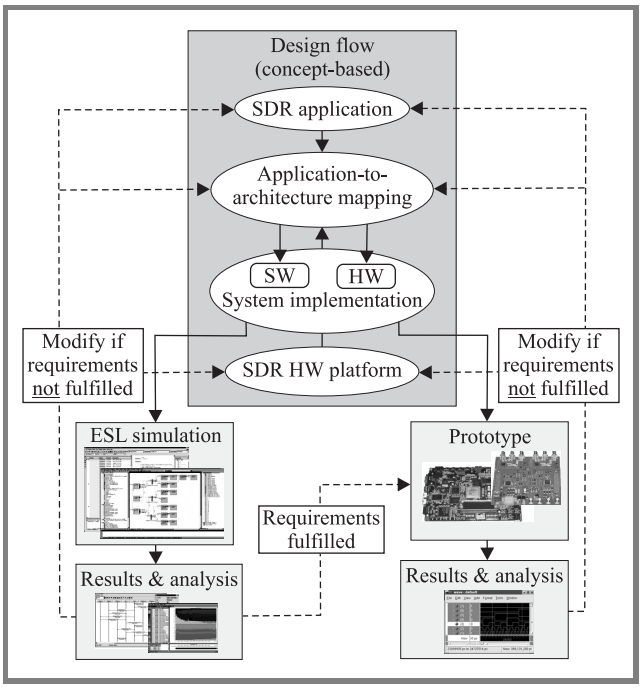


Fig. 3. System development flow.

(e.g., throughput, latency). Required modifications are applied to improve the mapping. ESL simulations can also be used within the SDR waveform development or hardware platform development cycle. In order to approach portability, the development iterations of waveform and hardware should be orthogonalized, ensuring that for example during the waveform development, only the waveform will be modified without relying on a specific mapping or hardware platform. If all requirements are fulfilled, developers can leave the ESL domain and can go into the phase of prototyping. This is usually at a point, where the bug detection rate decreases in a saturation process [9]. Bug detection now requires extensive simulation at low abstraction level (full hardware details). Using a prototype implementation can significantly speed up the bug detection process [10]. Possible issues occurring on the prototype are:

- Undetected software and system errors. Some errors only occur after extensive testing (coverage issues).
- Errors in the design of hardware components that were not captured in the system level model (modeling errors).
- Errors that only occur in a real time environment, for example interaction with analog frontend/analog components.

Therefore, debugging capabilities of the SDR prototype with feedback loops to the simulation are inevitable to allow developers to identify and fix such errors efficiently. In the following section we will highlight these problems based on an exemplary waveform and an FPGA prototype. In Section 4 a solution will be then introduced that allows to cope with the debugging issues.

3. SDR development flow

The WDE concept presented in Section 2 has been realized as a development flow which will be discussed in the following. First, the SDR application waveform used in the study will be shortly introduced followed by an exemplary SDR hardware platform. The system implementation, simulation and prototype will then be discussed with respect to the debugging and error analysis capabilities.

3.1. SDR application – example waveform

The waveform utilized in this study has been selected in order to keep the design complexity reasonable, while putting the focus on the investigation of the SDR development flow aspects. Therefore, the waveform does not include channel coding, interleaving, pilot insertion, etc. The waveform implements a differential quaternary phase shift keying (DQPSK) modulation scheme, pulse shaping is done by a Root-Raised-Cosine filter (roll-off factor 0.22, oversampling factor 8), resulting in a symbol rate of 5 MHz at a sampling rate of 40 MHz. The center frequency is in the 2.4 GHz band, the receiver input is a signal at an intermediate frequency (IF) of 10 MHz, mixed down to the base-band by a digital down converter (DDC).
For data transmission purpose, a simple frame structure consisting of 128 symbols has been defined. The payload data is a small black and white picture of 160×160 pixels in size. The picture is transmitted one line per frame with the line number transmitted at the beginning of the frame followed by the pixel data.
The transmitter will not be discussed in the following since it consists only of a DQPSK modulator and a pulse form filter. It is a pure VHDL implementation due to the simplicity of the given waveform. The receiver topology of this SDR application is given in the block diagram in Fig. 4. The signal is received as complex values at an IF of 10 MHz converted to base band by the DDC block and filtered by the matched filter (MF) block. The further processing only requires timing synchronization for symbol detection and a non-data-aided phase synchronization. The demodulator is directly followed by the frame synchronization and payload data decoder.

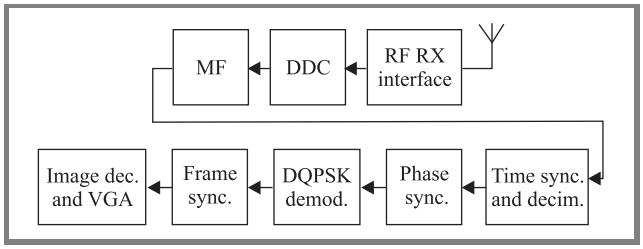


Fig. 4. Receiver block diagram for demonstration waveform.

In future, this waveform can be easily extended or replaced depending on the focus of investigations which in our case will be the SDR development flow aspects rather than on the waveform itself.

3.2. Example SDR hardware platform

The main parts of the SDR receiver are software (C-code), only parts with high computational requirements are realized in hardware (VHDL). Therefore, the system basically consists of two processing elements as depicted in Fig. 5 (a MicroBlaze general-purpose RISC processor [11] and a hardware accelerator). The MicroBlaze processor has been selected at this point in time for its ease of use and debugging capabilities. It does not allow real-time operation (scaled performance only). In future, a more suitable, real-time capable processor will be used.

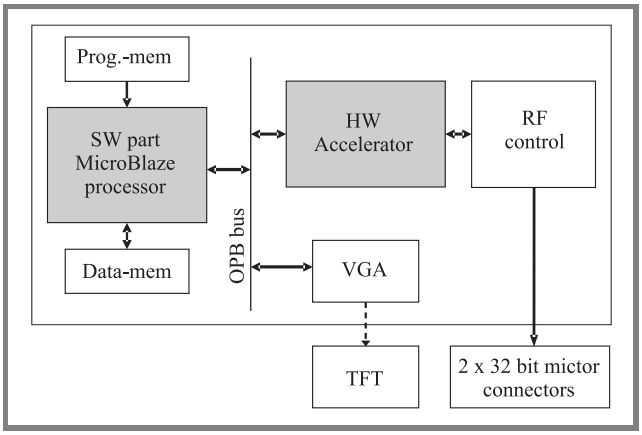


Fig. 5. Sample SDR platform used in this study.

The processor is connected to the peripherals (e.g., VGA port) by buses and IP cores. The hardware accelerators implement DDC and a MF. The basic IP core for accessing the SRFC RF interface has been provided by Signalion [12] and extended for the use in the MicroBlaze environment.

3.3. Mapping

With the SDR application and HW platform description developers can accomplish the mapping phase according to the concept. Each task of the SDR application has to be mapped onto a processing element (PE) of the underlying SDR HW platform. For this exemplary SDR platform, the mapping step is rather simple since only the MicroBlaze processor and dedicated hardware accelerators exist. Therefore, the mapping allocates the RF RX interface, the MF and DDC block as 1 : 1 mapping to the dedicated hardware accelerators, whereas all other parts have a temporal mapping as software tasks onto the MicroBlaze processor.

The system implementation can then be generated from the defined mapping as depicted in Fig. 3. Such a system implementation can be an ESL simulation or an FPGA prototype system, depending on the intended use case, e.g., whether a hardware implementation or a system model on a higher abstraction level will be investigated.

3.4. Simulation environment

The simulation of the complete SDR system is performed in the ESL domain. This allows the developer to test

the interaction of all components at the same time and at different abstraction levels of the implementation.
System simulation and abstraction provide many advantages in ESL design. However, there are also issues with this approach: raising the abstraction level introduces inaccuracy. Therefore, a successful detection depends on the type of error and on the selection of an appropriate abstraction level. In our simulation, the components of the hardware platform have been modeled with the abstractions described in the following. More accurate models can be selected when required.

- The MicroBlaze processor is represented by an instruction accurate simulation model.
- The DDC and MF components have been modeled by SystemC blocks [13]. In this implementation, their latency was not included in the system simulation models, although this is possible in general.
- The RF interface has been reduced to a clocked source of complex valued channel data. The control interface has been reduced to the functional minimum.

However, the use of the system simulation results in several important advantages:

- **Visibility**. The developer has control over the complete internal state of all components and can trace the states of special interest. The possibilities for visualization of such traces for a time instant or over time are manifold.
- **Synchronicity**. The system can be halted synchronously. Therefore, no component will continue to run while the state of another is being investigated, guaranteeing a consistent debugging environment over time.

The following sections describe the FPGA prototype setup, the hardware implementation of the SDR platform its debugging capabilities.

3.5. SDR hardware implementation – prototyping

Prototyping platform setup. The hardware prototyping platform is depicted in Fig. 6. It comprises an off-the-shelf Xilinx Virtex 4 FPGA development board, namely the ML402 board and a commercial RF-interface (SRFC) from Signalion [12] with two channels in the 2.4 and 5.0 GHz bands. The ML402 board provides on-board memories as well as a series of common interfaces. For debugging purposes, a standard JTAG interface is available which connects (among other chips) to the FPGA. Designs within the FPGA can connect to this JTAG chain, for example the MicroBlaze processor core. The Signalion SRFC RF frontend is connected to the 64-bit general purpose header of the ML402 board via an adapter card which provides access to the channel data and additional 22 signals via two mictor connectors. Sampling rates can be coarsely adjusted between 8 MHz and 80 MHz. In this setup, a sampling rate of 40 MHz is used.

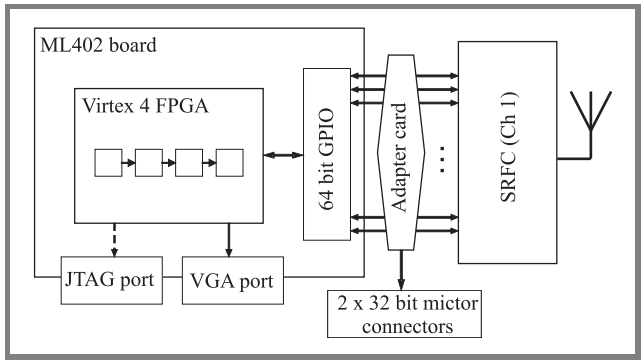


Fig. 6. Hardware prototype platform.

The SDR hardware platform implementation. The hardware platform has been set up with the Xilinx EDK development kit from a set of basic IP blocks connected by buses. This implementation flow clearly follows the proposed concept, the only gap that needs to be bridged later is the translation from the system simulation configuration into an EDK project configuration. The software binaries already used in the system simulation can be re-used without any changes for running the receiver prototype. The RF interface is connected by an IP core provided by Signalion [12]. The hardware acceleration blocks for the down conversion (DDC) and the matched filter have been implemented by IP cores generated from the Xilinx IP core library. FIFO buffers connected to the buses guarantee that blocks of continuous samples will be received by the processor.

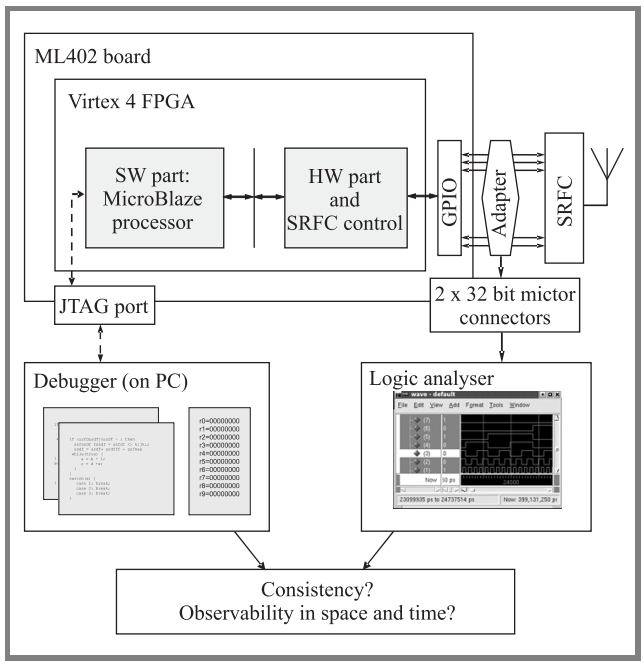


Fig. 7. Prototype debugging: consistency and visibility.

Prototype debugging. The debugging options available on the prototype platform used in this study are shown in Fig. 7. One traditional way of debugging is the hardware access via a JTAG interface [14]. Peripherals not connected to the JTAG chain, e.g., the RF interface, will continue to produce data, for example in case the processor is being debugged via JTAG, stimulus data from the RF frontend will be lost. Therefore, the time span for investigating such a prototype is very limited, analysis of past or future states is almost impossible or includes the risk of being inconsistent due to non synchronized IP blocks.
In order to deal with the temporal visibility, the requirements are high: investigating 64 bit at only 40 MHz results already in 320 MByte/s. The use of a logic analyser allows to record at this rate and at least solves the problem of analysing future and past by selecting the trigger position relatively to the recorded data carefully.
The traditional top-down approach which has also been part of the proposed WDE concept (see Section 2) requires an extension in order to allow the analysis of errors found in a complex prototype system, such as an SDR. This extension by feedback loops of stimulus data will be discussed in the following section.

4. Stimulus feedback to higher abstraction levels

As demonstrated, it is desirable to analyze errors, which were detected on the prototype, in the ESL simulation. However, there is a significant issue. It is neither sufficient nor possible to trace and copy the SDR hardware state to the system simulation due to limited visibility, possible differences in accuracy and the size of the complete state information. One option is to feed back the essential stimulus data from the prototype to the system simulation.

4.1. Requirements on stimulus recording and feedback

Typically, there is a delay between the occurrence of an error and the observation of its effect. Therefore, a thorough analysis of the observed malfunction and the recent processing steps is necessary for debugging. Viewing past events usually is not possible on a real HW implementation, but it is feasible to record the stimulus data which caused the error and to use this as input for the ESL simulation. In order to allow such feedback of stimulus data the following requirements have to be fulfilled:

- **State reachability**. The stimulus data must be suitable to drive the system into the state where the bug can be detected and analyzed. This also poses requirements on the accuracy of the system simulation and the interfacing to the lower abstraction levels.
- **Looking into the past**. The stimulus data must cover a specific amount of the time before the bug is detected. This puts high requirements on the recording technology in order to store a significant amount of stimulus data before detecting the bug.
- **Stimulus recording**. The stimulus data can arrive at high data rates. Interfaces such as serial communications, USB or hard disk are not capable to deal with such data rates.
- **Stimulus import**. Seamless stimulus import is key for each simulation setup. This is basically a requirement for the simulation interfaces and optional data format conversion.

A question is, how much stimulus data before the trigger point is required to reach the same system state that includes the underlying error. Because of recursive structures it may in principle require collection of data starting with the last known state (e.g., after reset). However, in the signal processing domain, many subsystems implement either stream like processing without recursion (e.g., FIR filters or buffers) or have a limited number of recursions. Therefore, such systems can be steered into a desired state by appropriate choice of history length. This will most probably not apply for layers above the physical layer. However, internal states, e.g., on layers 2 and 3 are changing at much lower rates, making state recording feasible. Therefore, future extensions will have to consider this mix of stimulus and state recording for higher layers.

4.2. Stimulus feedback implementation

Since the SDR application depicted in Fig. 4 does not contain loops on the task level and no recursion inside the tasks, the internal state of the SDR application only depends on a limited number of recent input samples received from the RF frontend. Therefore, recording the digitized channel data at the interface to the SRFC RF frontend is sufficient. The amount of data that needs to be recorded before a bug is detected depends mainly on the latencies of filters and FIFO buffers.
For recording stimulus data, the original adapter card connecting the GPIO connector on the ML402 board to the SRFC RF frontend had been extended by two mictor connectors, providing a total of 64 bit of parallel data. Since the samples on the RF RX channel have been selected, stimulus data is being recorded at a sample rate synchronous to the sampling clock. The SRFC interfaces only occupies 42 of the 64 bit on the mictor interface, leaving further 22 free bits for steering the recording trigger or additional stimulus data.
The data is recorded by a logic analyzer connected to the mictor connectors on the adapter card. In this setup, the logic analyzer samples the data with the RF RX sampling clock provided by one signal on the mictor connectors. Thus the logic analyzer is synchronous with the SDR receiver implementation and the recorded data can be used as stimulus within the system simulation.
To drive the ESL simulation into the correct state, the stimulus data before a bug detection has to be recorded. Modern logic analysers offer a trigger option, that allows for the setting of the trigger position at any point in time

within the later recorded data. Of course, the trigger condition has to be set up manually. This requires a designer to drive the free debug signals on the mictor connectors accordingly (Fig. 8).

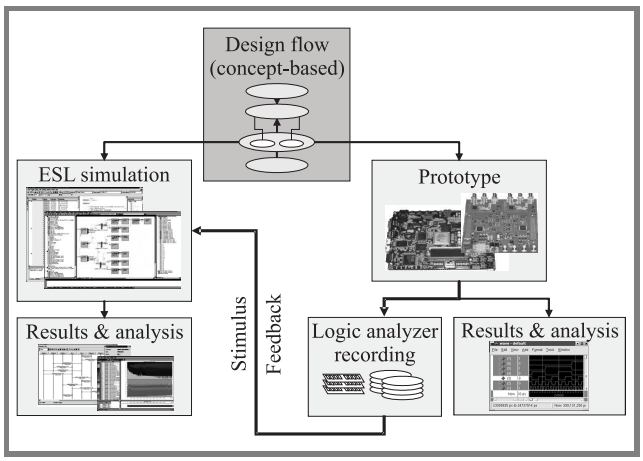


Fig. 8. Stimulus feedback loop realization.

The recorded data has been imported in ESL simulations and even into Matlab simulations. This step basically consists of the implementation of suitable import filters. The data can be taken expressively or it might require the conversion, e.g., to fixed/floating point number representations like in Matlab. For the given example SDR system, importer filters for both Matlab and the ESL simulation have been developed.

4.3. Design process experience

This stimulus feedback setup has been used during the initial setup of the SDR hardware platform with the intention to be used in future more extensively. Nevertheless, the feedback setup has already been valuable during the development steps of the SDR receiver system presented above. On this platform, several basic algorithmic implementations have been tested, for example, the implementations of estimations and corrections for timing, phase and frequency errors.

One of the most helpful features of an ESL simulation using the recorded data has been the aspect of visualization. Depending on the type of data (e.g., complex values) a suitable visualization (e.g., I/Q diagram) can be generated from any trace of internal states, data transfers on buses, etc. Therefore, the data analysis is virtually unlimited and gives valuable support to the SDR developer.

When testing the first implementations of a basic frequency error estimation and correction, the synchronization result has been visualized on the VGA port of the prototype system first. On the prototype, the frequency synchronization result was hardly recognizable while the system simulation seemed to work well with a generic channel input. After some investigation the input stream sampled within the FPGA was recorded by the logic analyzer and fed back into the ESL simulation. The errors could be reproduced and visualized, this time even with Matlab. The key point in this case has been the visualization which uncovered occasional phase shifts by 90°. The investigation showed that this error has been caused by a combination of the DDC hardware block together with the erroneous clocking setup in the FPGA which created an occasionally mis-sampled data stream. The interesting aspect is the fact, that although the error cause has been in very low level hardware, the feedback flow proved to be highly efficient for error tracking and analysis.

Other errors have been localized in the software implementation, e.g., during the initial implementation of the timing synchronization. Here, it has been extremely helpful to debug the software implementation while comparing the outputs visually with the results of the Matlab simulation for the same input data.

Therefore, the extension of the WDE development concept by stimulus feedback loops results in valuable benefit for already small scale SDR systems. It will most probably be inevitable for highly complex future SDR systems. Future extensions will need to include state recording at much lower rates for layers above the physical layer.

5. Conclusion and outlook

In this paper we proposed an extension to our WDE concept that feeds back stimulus data from a hardware implementation such as a prototype back to the system simulation. This allows the more efficient error analysis. We believe, that such a feedback concept is essential for the development of future complex SDR systems.

In future, this concept will be used in our SDR development approaches based on the WDE concept. Extensions will need to include state recording at much lower rates for layers above the physical layer. So far, the steps of the proposed development flow had been realized mainly independently from each other. Therefore one important point for future development is the realization of a seamless prototype work bench for SDR development ranging from the waveform's specification down to the implementation. Furthermore, with the help of this tool chain we will investigate more realistic SDR systems and communication standards like, e.g., the MIL-STD-188-110B. Additionally we will investigate more in depth the key issue of portability for SDRs.

Acknowledgements

This research project was performed under contract with the Technical Center for Information Technology and Electronics (WTD-81), Germany.

The authors would like to thank J. Holzer, C. Hatzig, S. Hartmann and H. Siegmar of this Center for inspiring discussions.

References

[1] T. Kempf, E. M. Witte, V. Ramakrishnan, G. Ascheid, M. Adrat, and M. Antweiler, "An SDR implementation concept based on waveform description", *Freq. J. RF-Eng. Telecommun.*, vol. 60, iss. 9–10, pp. 171–175, 2006.

[2] "Software communications architecture (SCA) specifications V2.2", JTRS, http://sca.jpeojtrs.mil

[3] E. D. Willink, "Waveform description language: moving from implementation to specification", in *IEEE Milit. Commun. Conf. MILCOM 2001*, Vienna, Virginia, USA, 2001, vol. 1, pp. 208–212.

[4] M. S. Gudaitis and R. D. Hinman, "Practical considerations for a waveform development environment", in *IEEE Milit. Commun. Conf. MILCOM 2001*, Vienna, Virginia, USA, 2001, vol. 1, pp. 190–194.

[5] M. Grant, B. Bailey, and A. Piziali, *Electronic System Level Design and Verification*. San Francisco: Morgan Kaufmann, 2007.

[6] T. Kempf, M. Adrat, E. M. Witte, V. Ramakrishnan, M. Antweiler, and G. Ascheid, "On the feasibility of implementing a waveform application onto a given SDR platform", in *Milit. CIS Conf. 2006 MCC 2006 (formerly NATO RCMCIS)*, Gdynia, Poland, 2006.

[7] T. Kempf, E. M. Witte, V. Ramakrishnan, G. Ascheid, M. Adrat, and M. Antweiler, "A workbench for waveform description based SDR implementation", in *Softw. Defin. Radio Tech. Conf.*, Denver, USA, 2007.

[8] "Platform studio and the EDK", Xilinx, http://www.xilinx.com/ise/embedded_design_prod/platform_studio.htm

[9] Y. Malka and A. Ziv, "Design reliability – estimation through statistical analysis of bug discovery data", in *Proc. Des. Automat. Conf. DAC'98*, San Francisco, USA, 1998.

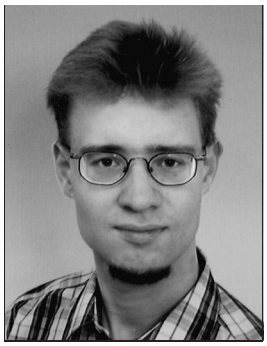
[10] K. Morris, "Debug dilemma – simulate or emulate?", *FPGA Programm. Log. J.*, Jan. 2005, http://www.fpgajournal.com/articles_2005/20050111_debug.htm

[11] "Microblaze processor reference guide", Xilinx, http://www.xilinx.com/ise/embedded/mb_ref_guide.pdf

[12] "Prototyping the wireless future", Signalion GmbH, http://www.signalion.de

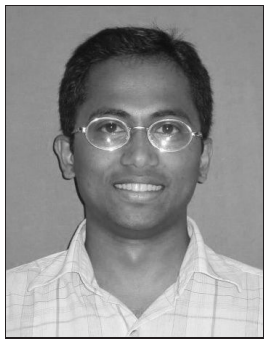
[13] T. Grötker, S. Liao, G. Martin, and S. Swan, *System Design with SystemC*. Norwell: Kluwer, 2002.

[14] "Standard Test Access Port and Boundary-Scan Architecture", IEEE Std. 1149.1, 2001.



Ernst Martin Witte received his Dipl.-Ing. degree in electrical engineering in August 2004 from the Institute for Integrated Signal Processing Systems, RWTH Aachen University, Germany, where he is currently pursuing the Ph.D. degree. Currently, his research in the area of software defined radio focuses on architecture exploration, implementation and prototyping of application specific instruction-set processors.
e-mail: witte@iss.rwth-aachen.de
Institute for Integrated Signal Processing Systems
RWTH Aachen University
Templergraben 55
D-52056 Aachen, Germany

Torsten Kempf received his Dipl.-Ing. degree in electrical engineering from RWTH Aachen University, Germany, in December 2003. In 2004 he joined the Institute for Integrated Signal Processing Systems and is currently pursuing the Ph.D. degree. Currently his research topics are multiprocessor system on chips, electronic system level design and software defined radios.
e-mail: kempf@iss.rwth-aachen.de
Institute for Integrated Signal Processing Systems
RWTH Aachen University
Templergraben 55
D-52056 Aachen, Germany



Venkatesh Ramakrishnan received his M.Sc. degree in information and communication engineering from University of Karlsruhe, Germany, in 2004. He is currently pursuing the Ph.D. degree at the Institute for Integrated Signal Processing Systems, RWTH Aachen University, Germany. Currently, his research in the area of software defined radio focuses on the implementation of waveform and prototyping.
e-mail: ramakris@iss.rwth-aachen.de
Institute for Integrated Signal Processing Systems
RWTH Aachen University
Templergraben 55
D-52056 Aachen, Germany

Gerd Ascheid received his Dipl.-Ing. and Dr.-Ing. degrees in electrical engineering (communications eng.) from RWTH Aachen University, Germany. In 1988 he started as a co-founder CADIS GmbH. The company has successfully brought the system simulation tool COSSAP to the market. In 1994 CADIS GmbH was acquired by SYNOPSYS, a California-based EDA market leader where his last position was Senior Director (executive management), wireless and broadband communications service line, synopsys professional services. Since April 2003 he is the Head of Institute for Integrated Signal Processing of the RWTH Aachen University (as successor of Prof. Heinrich Meyr). He is also the Chairman of the cluster of excellence in "Ultra-high speed Mobile Information and Communication (UMIC)" at RWTH Aachen University.

e-mail: ascheid@iss.rwth-aachen.de
Institute for Integrated Signal Processing Systems
RWTH Aachen University
Templergraben 55
D-52056 Aachen, Germany

Marc Adrat received his Dipl.-Ing. degree in electrical engineering and the Dr.-Ing. degree from RWTH Aachen University, Germany, in 1997 and 2003, respectively. From January 1998 to March 2005, he was with the Institute of Communication Systems and Data Processing at RWTH Aachen University. His work was focused on joint/combined source-channel (de)coding for wireless communications with the main focus on iterative, turbo like processes. Since April 2005, he is with the Research Establishment for Applied Sience (FGAN FKIE/KOM) in Wachtberg. His current research interests include software defined radio, cognitive radio, (military) waveform design as well as concepts for a waveform description language.

e-mail: adrat@fgan.de
Research Institute for Communications,
Information Processing, and Ergonomics (FKIE)
Research Establishment for Applied Science (FGAN)
Neuenahrer st 20
D-53343 Wachtberg-Werthhoven, Germany

Markus Antweiler received his Dipl.-Ing. and Dr.-Ing. degrees from the RWTH Aachen University, Germany, in 1986 and 1992, respectively. The focus in his industry career was on system design and verification of digital communication transceivers and implementation in application specific integrated circuits and field programmable gate array technology. Projects he has worked for were in the area of digital modulation/demodulation and coding/decoding for satellite communication, microwave links, cellular and wireless communication systems. In 2004, he joint the Research Institute for Communications, Information Processing, and Ergonomics of the Research Establishment for Applied Science (FGAN e.V.) in Wachtberg, where he is heading the Communication Systems Department. His current interests are now on tactical communications with focus on mobile ad hoc networks, security, software defined radios and reconaissance of radio systems.

e-mail: antweiler@fgan.de
Research Institute for Communications,
Information Processing, and Ergonomics (FKIE)
Research Establishment for Applied Science (FGAN)
Neuenahrer st 20
D-53343 Wachtberg-Werthhoven, Germany