# MATLAB BASED CODESIGN FRAMEWORK FOR WIRELESS BROADBAND COMMUNICATION DSPS

*Volker Aue, Johannes Kneip, Matthias Weiss, Michael Bolle, and Gerhard Fettweis*

Systemonic AG, Am Waldschlösschen 1
D-01099 Dresden, Germany
info@systemonic.de

## ABSTRACT

We provide an overview of a novel MATLAB™ based Hardware-Software design flow that has been applied to the design of a platform based SoC for the HiperLAN/2 and IEEE 802.11a wideband wireless communication standards. Starting from a high-level algorithmic description, the MATLAB™ environment serves as a "golden model" and universal, cycle-true testbench for the embedded hardware and software implementation. A universal interface concept allows the exchange of modules with different abstraction levels in a cosimulation. This way, a high confidence level for the design validation is achieved, and both design and validation time is substantially reduced.

## 1. INTRODUCTION

With the ongoing trend to always shortening product life spans and development cycles the efficiency of the development and validation process of application-specific standard processors (ASSPs) and their firmware is of increasing importance. The complexity of today's communication systems does not allow the application of the classic development and validation environment consisting of models of different abstraction levels and a customized testbench for each level.

In this paper we present a case study for a platform based communication processor design targeting HiperLAN/2 and IEEE 802.11a applications. The processor consists of a programmable SIMD core and a number of dedicated building blocks. For this ASSP, a MATLAB™ based design flow has been developed that allows the codesign and covalidation of the algorithmic model, the embedded hardware and firmware in a single test environment and the same test data and channel models.

The following prerequisites had to be met when the codesign/covalidation environment was developed:

- The environment should provide the information necessary to allow the partitioning between the programmable core and dedicated blocks. This includes a first assessment of the algorithmic complexity, but also the influence of algorithmic changes onto the system behavior and complexity.

- After partitioning the parallel development of hardware and software requires very close interaction. Interoperability and interfacing of hardware and software modules have to be checked on any stage of modeling. Model layers include the algorithmic MATLAB™ model, a System-C software model of the ASSP, RTL and netlist models and the embedded firmware executed by the hardware models.

- Effects caused by embedded implementation, e.g. word length effects and delays caused by processing were to be taken into account. Bit- and cycle-true simulation results need to be fed back to the system model to assess (and probably correct) these effects. This step must be possible for both modules implemented in hardware and modules implemented in software.

- The equivalence of hardware and software models or implementations on different levels of abstraction has to be validated by simulations with realistic data. This has proven to be a nontrivial task, since timing behavior and output results may differ between the levels due to word length and timing effects discussed above.

- Input and output reference data gained from the high level simulation model should be accessible from any implementation or modeling level. Ideally, test bench frameworks should also be reusable between the levels to reduce overall engineering effort.

MATLAB™ has been chosen as basis for the platform because it is wide spread, shows a high level of interactivity and provides interfaces for script based execution and exchange of data with other applications.

This paper discusses several aspects of our codesign methodology with a focus on the HiperLAN/2 and IEEE

802.11a application case study. We start with a short description of the platform approached used for the processor. Section 3 discusses the design flow that leads from a high-level algorithmic description via system partitioning to an optimized embedded implementation. Finally, Section 4 presents the results of the application of our methodology for the case of the HiperLAN/2 ASSP.

## 2. THE OnDSP™ PLATFORM

To combine high performance and economy of a platform, the OnDSP™ platform allows the generation of parallel DSP derivatives, but also the corresponding code development and simulation tools. Derivatives are widely configurable in terms of the memory system configuration, arithmetic functionality and word length, communication structure and interfacing.

Fig. 1 shows an example of a core generated from the framework for the HiperLAN/2 application. It consists of a program control unit (PCU) and, in this case, eight parallel data paths. Data paths (DPUs) are *single instruction multiple data* (SIMD) controlled and consist of a RISC like register file and an arithmetic unit, in this example containing a 40 bit ALU, a barrel shift unit, and a 17-by-17 bit MAC with 40 bit accu. The DPUs have access to a wide memory via an interconnection unit that also allows broadcast and exchange of data between the data paths. Special instructions have been introduced to speed up important algorithms like FFT, filters and QAM modulation and demodulation. The core itself is supplemented with application specific building blocks for the algorithmic parts of the system. This may either be algorithms that do not require the flexibility of the programmable platform or where a dedicated implementation promises significant advantages in area or power consumption compared to a software based implementation.

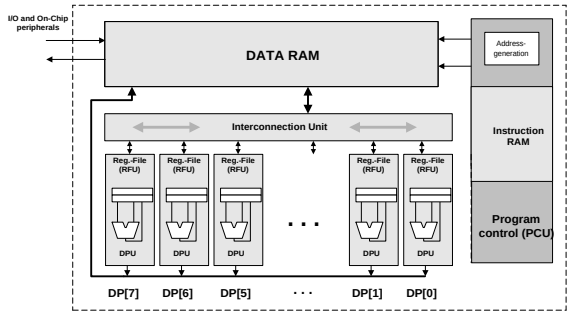


**Fig. 1**. Example for a core with 8 parallel data paths derived from the OnDSP™ framework.

The programming of the DSP core starts from a high level language description of the algorithm to be implemented. This may either be C, C++, or a Matlab reference implementation. The target is an optimized assembly code generation, utilizing the DSP's parallel data paths and the VLIW instruction word.

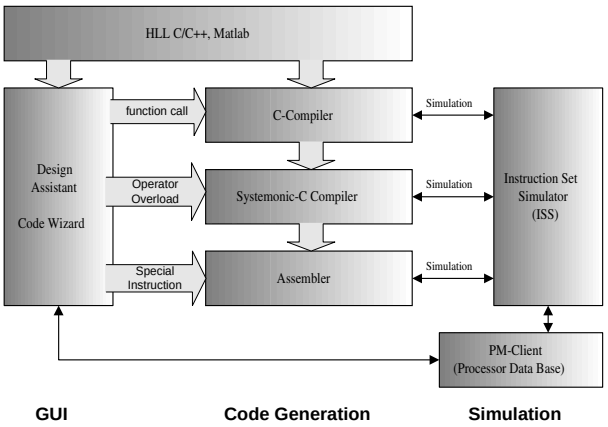


**Fig. 2**. Overview of the OnDSP™ code generation process and tool environment.

Fig. 2 provides an overview of the tool hierarchy. The main tool is the *Design Assistant*, an integrated framework for all underlying development tools and the corresponding processor architecture data base. This framework not only helps to generate robust firmware development tools (another approach is given in [2]) but also automatically derives optimized core algorithms. Programming of the DSP core is mainly done in *Systemonic-C*, a C++ class library reflecting both the DSP's instruction set and parallelism to allow the application of known optimization techniques (e.g. loop transformations) [5].

## 3. DESIGN FLOW

The DSP software code development process described in Section 2 is integrated in our MATLAB™ based system design flow. After the target *system on chip* (SoC) device has been identified, at first a model of the wireless system, of which the SoC device is a part of, is created. The design flow consists of the following steps.

1. Create a system model according to the system standard specification.
2. Refine the model of the SoC device.
3. Hardware – software partitioning.
4. Hardware – software codevelopment using the OnDSP™ platform.
5. Integration of hardware and software, and co-simulation.
6. Chip and board level tests.

The first system model is created in MATLAB™ relying mostly on MATLAB™ script functions. Floating point precision is used at this modeling stage. Modeling the system begins with describing the transmitter signals, where the transmitter signals are typically defined in the specification. Either system standard channel models (AWGN) are used or models created from real measured data. Using the channel models, the receiver demodulation routines can be refined, and the synchronization algorithms, i.e., acquisition and tracking algorithms can be developed. After the system has been modelled and is well understood, the SoC algorithms are further refined. This step includes a conversion from floating to fixed point, and an algorithm optimization with respect to expected memory consumption, and computational complexity. In particular, the algorithms are modified to explicitly expressed parallelism that later on can be exploited by the OnDSP™ signal processor architecture. For fixed point modeling, a MATLAB™ DSP fixed point class library has been developed. This library contains special data types for 8, 16, and 32 bit data types to model 16 bit DSP fixed point behavior with single and double precision, and split mode capabilities. An additional 40 bit type exists to model accumulator behavior. Fractional integer classes such as Q15 are also introduced. Operators are overloaded to ease programming. Converter functions from one class to another including conversion to floating point are also defined. The DSP class library facilitates floating point to fixed point conversion of the target algorithms, since fixed point arithmetic can be easily expressed. Even though the algorithms are modified from the original floating model, the interfaces of the SoC model are kept. This way, the fixed point model can always be simulated or compared against the floating point model, or it can be simulated in the context of the entire system.

The next step in the SoC design is partitioning of the SoC device into hardware and software. Here, required computational power and memory are estimated from the MATLAB™ SoC model. The OnDSP™ core parameters are entered into the processor data base, and a corresponding ISS is generated. The DSP core hardware-software codesign flow described in Section 2 fits well into the design flow, since the ISS is provided with interfaces to communicate with MATLAB™. MATLAB™ models can be substituted by their ISS equivalent counter parts.

For modeling dedicated logic, the fixed point algorithms from the previous development stage are converted into cycle true models. Here, we use the SystemC class library. Like the ISS models, the cycle true dedicated hardware models can interface with the MATLAB™ system simulation. This way, embedded software on the ISS, dedicated hardware models, and other MATLAB™ models can run all in the same simulation. After the routines have been individually validated, the MATLAB™ glue code is then converted into DSP assembly language.

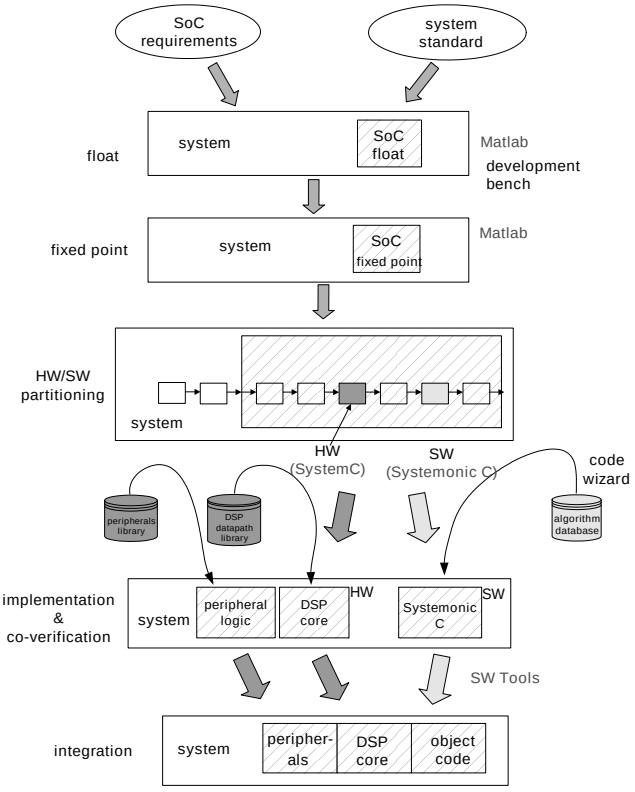


**Fig. 3**. Matlab based design flow.

Finally, the behavioral model of the entire hardware, or the RTL model are equipped with a test bench that can interface with MATLAB™. Furthermore these models can be substituted for the fixed point model, or compared against it.

The design flow described so far is shown in Fig. 3.

After the chip has been manufactured, the original system model is used to generate test signals, and to verify hardware and software. The test board is equipped with a special driver that allows for interfacing with the system simulation. Signal input patterns from the system simulation are sent to the board or can be downloaded into an arbitrary waveform generator. Digital output signals from the chip are uploaded to the simulator and validated against the simulated signals.

## 4. CASE STUDY HIPERLAN/2

The design flow described in Section 3 has been successfully applied to in the development of a baseband *application-specific signal processor* (ASSP) targeted towards HiperLAN/2 and IEEE 802.11a applications. The HiperSonic™ uses the OnDSP™ platform to perform all modulation, demodulation, and synchronization routines on a fully programmable DSP core. Furthermore, it includes

**Table 1**. Implementation data of the HiperSonic™ ASSP

| | |
|---|---|
| Process technology | 0.18 $\mu$m, 4LM standard cell |
| Core supply voltage | 1.8 V core, 3.3V I/O |
| Operating frequency | 120 MHz |
| Power consumption | less than 1 Watt |
| Transistor count | 7.6 Mio |
| Pin count | 176 |
| Packaging | MBGA 176, $15 \times 15$ mm$^2$ |
| A/D, D/A converter | 40/80 MHz sampling frequency, 10 bit resolution |
| Supported Standards | IEEE 802.11a, HiperLAN/2 |
| Maximum data rate | 60 Mbit/s |

on chip A/D and D/A converters, digital pre and post filters, and dedicated logic for channel coding and decoding as well as for basic MAC layer functions such as cyclic redundancy check (CRC) and data encryption. The chip can be used with a 5 GHz RF front-end chip and an off-the-shelf microprocessor, and an Ethernet or IEEE 1394 controller, to build modems for HiperLAN/2 office and home environments [1, 4], and wireless LAN modems compliant with the IEEE 802.11a standard [3]. By employing this codesign framework we were able to design the complex HiperSonic™ ASSP within a short time frame of several months. Details are given in Table 1.

At first, a system model has been created in Matlab. For HiperLAN/2 the model includes MAC frame generation, channel models that have been used by ETSI in the development of the HiperLAN/2 standard, and the impact of analog and digital filters. Sampling rate mismatch and frequency offsets are also included in the model. Modem algorithms have been converted to fixed point and optimized to reduce computational complexity while at the same time maintaining high performance. From the model, the system on chip hardware-software partitioning has been realized. It turned out that the OnDSP™ platform yields adequate performance to compute all modulation and demodulation routines on a single DSP core. This includes FFT/IFFT processing, frequency error correction, OFDM symbol (de)mapping, QAM mapping, and soft-bit QAM demapping. The DSP contains three different data memories. One block of RAM is used for storing variables and temporary data. The other two RAM blocks are used for DMA data input and output, respectively, in order to provide a high speed I/O interface to the DSP core. Channel coding is better performed in peripheral (side) units, since this involves bit-wise processing, and algorithms here need to be less flexible.

In the subsequent development, the DSP core and peripheral units are modelled and simulated at different levels of abstraction (see Section 2 and 3). All models are provided with an interface to MATLAB™.

Data that are read into Matlab are converted to Matlab matrices. In particular, memories are represented as matrices, where each element represents a 16 bit word. Thus, a convenient method is established to analyze and manipulate data. Using these interfaces, routines originally described in Matlab, are successively exchanged by their hardware or software equivalences. Simulations are run on different integration levels, and compared with the original fixed point model. This way, regression tests can be easily performed.

## 5. CONCLUSION

With the system driven design flow shown in this paper we present a homogenous design environment that spans from system-level simulation to embedded system implementation. Due to a highly automated design flow and interoperability of simulation models and pattern on differing levels of abstraction, design validation is made possible at any stage of the design process. Implementation effects caused by word length limitations and processing latency can be directly analyzed in a system level simulation. With the application of the design framework and a platform based processor architecture, only seven months were required from first system evaluation to tapeout of the HiperLAN/2 ASSP, called HiperSonic™. This demonstrates the potential of our approach to reduce design time and cost, while maintaining a high confidence level for the hardware and software quality.

## 6. REFERENCES


[1] European Telecommunications Standards Institute (ETSI). *Broadband Radio Access Network (BRAN); Hiperlan Typ 2; Physial (PHY) Layer*, Apr. 2000. TS 101 475.

[2] A. Hoffmann, S. Pees, and H. Meyr. ”a retargetable toolsuite for exploration of programmable architectures in SOC-design”. In *ICSPAT’99*, Orlando, USA, Nov. 1998.

[3] The Institute of Electrical end Electronics Engineers, Inc. *Supplement to IEEE Standard for Information technology — Telecommunications and information exchange between systems — Local and Metropolitan area networks — Specific requirements — Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed Physical Layer in the 5 GHz Band*, first edition, 1999. International Standard, ANSI/IEEE Std 802.11a.

[4] M. Johnsson. HiperLAN/2 — the broadband radio transmission technology operating in the 5 GHz frequency band. In *www.hiperlan2.com/site/home.htm*, 1999.

[5] M. Weiss, D. Fimmel, R. Merker, and G. Fettweis. Designing Performance Enhanced Digital Signal Processors Using Loop Transformations. In *PACT’98*, pages 90–94, Paris, France, 1998.