SDR WAVEFORM DEVELOPMENT: TOWARDS TOOL ASSISTED MAPPING AND EVALUATION IN THE NUCLEUS CONCEPT

Venkatesh Ramakrishnan[*], Marc Adrat[+], Torsten Kempf[*], Jeronimo Castrillon[*],
Gerd Ascheid[*], Rainer Leupers[*], Heinrich Meyr[*], Markus Antweiler[+]
[*)] Institute for Integrated Signal Processing Systems, RWTH Aachen University, Germany
[+)] Fraunhofer (FKIE), Wachtberg, Germany
{ramakrishnan@iss.rwth-aachen.de, marc.adrat@fkie.fraunhofer.de}

ABSTRACT

Software defined radio (SDR) solutions for mobile devices need to be energy efficient to extend battery life. At the same time, SDR solutions need to support easy portability of waveforms (WFs). To address these contradicting needs of efficiency and WF-portability, a novel concept, the Nucleus concept, has been proposed. The Nucleus concept, which is library based, provides flexibility in implementing a WF by making available efficient implementations for critical components of a WF. Due to the existence of a huge design space in mapping a WF to a hardware platform, tool assistance is highly beneficial, if not mandatory. An important step towards a mapping-tool is to analyze, understand and precisely capture the requirements of a tool in accordance with the Nucleus concept. This paper contributes a methodology for a tool-assisted mapping and evaluation in the Nucleus concept. Various information sources and the information that needs to be captured by the mapping-tool are identified. A two-level evaluation is proposed to enable fast convergence to an ideal mapping efficiently.

1. INTRODUCTION

New generations of mobile devices need to implement multiple waveforms (WFs) with different transmission schemes, data-rates, etc. on a single hardware (HW) platform. Software defined radio (SDR) based mobile devices promise to meet such needs. A WF represents a complete wireless standard with several modes. Energy efficiency is a key requirement in SDRs, due to its impact on battery life. Therefore, one of the most promising approaches to implement SDRs is a heterogeneous multiprocessor system-on-chip. At the same time, portability of a WF is another contradicting requirement. One way to fulfill both these requirements is to use a library based approach for WF-development. Efficient implementations, from a library, of some/all components of a WF, can reduce porting effort of a WF to a HW platform. One of the main limitations in traditional library based WF-development approaches is the specificity to one WF or one HW [1] [2]. For example, the library in [1] is specific to one WF, WiMax. In our previous work, we have introduced a new concept for developing WFs [3] to overcome this limitation.

A new classification of library elements, called Nuclei, has been proposed targeting on reusability, portability and implementation efficiency of the WFs [3]. Figure 1 illustrates the process of WF-development using the Nucleus concept. A **Nucleus** is defined as a critical, demanding, algorithmic kernel that captures common functionalities within and/or among WFs. A **Flavor** is an optimized and efficient implementation of a Nucleus. A library that consists of Nuclei, the **Nucleus-library,** is used for constructing a WF by assembling Nuclei (Figure 1). A **non-Nucleus** is a computation-light kernel in a waveform, e.g. a modulator.

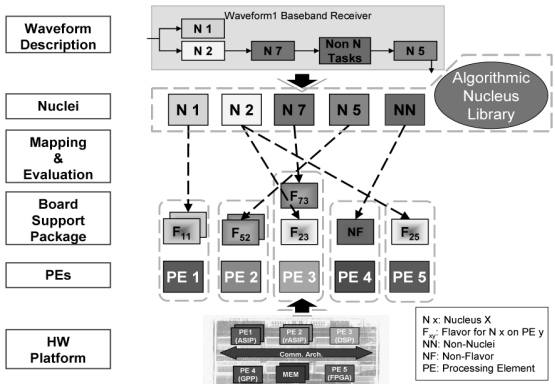

Figure 1. WF-development using the Nucleus concept [3]

Flavors could be assembly codes, e.g. for a DSP and/or IP cores in HW, e.g. for a FPGA. Hence, a Flavor is a bundle of the processing element (PE) and low-level software (if applicable). This enables to describe a WF

This research project was performed in the UMIC (Ultra High-Speed Mobile Information and Communication) research centre under the support of the Technical Center for Information Technology and Electronics (WTD-81), Germany.

independent of the HW, as a platform independent model (PIM) [3]. Due to the HW-bundled Nucleus low-level implementation, a WF can be mapped efficiently to a HW and a platform specific model (PSM) of the WF can be obtained. Upon standardization of the Nucleus library, vendors can provide efficient implementations of some or all of the Nuclei in the form of a board support package (BSP).

One of the main challenges in realizing the Nucleus concept, mapping with tool-assistance, is addressed in this paper. By mapping, we mean the process of spatially and temporally distributing the functionality of a WF in an efficient way onto the resources of a HW such that the WF-constraints are met. Evaluation, where constraint checks are done, can be considered as an integral process in mapping. For clarity reasons, it is treated as a separate process in this paper, whenever explicitly mentioned. Examples have been provided, at appropriate sections, for explaining the mapping methodology. Key contributions of this paper are:

- Important information, and its sources, needed for a tool-assisted mapping is provided.
- A methodology of tool-assisted mapping and evaluation for the Nucleus concept is proposed.

2. RELATED WORK AND MOTIVATION

Clever mapping concepts have been based on constructing models of the WF and HW separately in accordance with the model driven architecture [4] [5]. However, modeling in terms of PIM and PSM varies among concepts and hence the mapping requirements also vary. For example, [4] considers separate models for functionality and algorithms-for-functionality of a WF. In our Nucleus concept, both models will be integrated into a single model since the components themselves are algorithm-based. Authors in [5] have proposed a mapping process onto parallel processor architectures. Tools for mapping are also considered. However, here again, modeling is different when compared to our Nucleus concept. For example, a Nucleus model in a WF-description (functional model in [5]) consists of more information, like properties, that is used for our mapping process. By properties, we refer to performance related properties like bit error rate (BER), latency, throughput, etc. The HW model in [5] is the target HW that is being built. In our case, the HW model is given. Compared to [4] and [5], partitioning and code generation is also minimal because of using Nuclei library for WF-description and BSP for WF-implementation. More differences in the Nucleus concept, when compared to the existing solutions, motivate the need for a new mapping methodology and tools.

The fundamental difference in the Nucleus concept, when compared to other approaches [4-6], is that, mapping of a WF-functionality is from a Nucleus to its corresponding Flavor and not directly to a PE (Figure 1). With a rich BSP, there can be several Flavors for one Nucleus on: a) one PE or b) different PEs. Also, Flavors for different Nuclei might be available on the same PE. Our recent work [7] has shown that Flavors demonstrate wide trade-offs in terms of properties, e.g. BER. Furthermore, several constraints of a Flavor can introduce additional overhead in terms of properties and can have a huge influence on the performance of a WF, e.g. input data-width on memory. In addition, parameters of a Nucleus, e.g. processing mode, influence the properties of a WF, e.g. throughput [7]. By parameters, we refer to configuration parameters like input data-width, scaling schedule, processing mode, etc. Therefore, careful selection of a Flavor and its parameters for a Nucleus is one of the key necessities for an efficient WF-implementation.

Apart from the Flavor itself, it is important to understand the other factors that influence the performance of a WF-implementation. One such factor is the memory hierarchy. In general, a Flavor needs additional resources like memory to store data, e.g. coefficients. The distribution of such data for a Flavor in a PE is one of the influencing factors. This can be explained using the memory hierarchy of TMS320C64x, from TI, as an example [8].

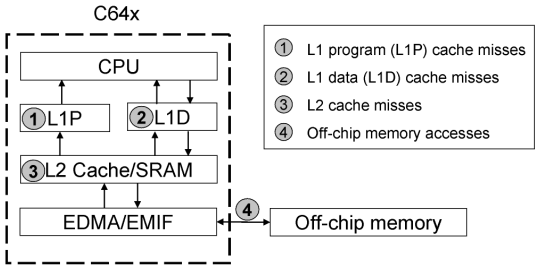

Figure 2: Memory hierarchy and potential overhead [8]

Figure 2 illustrates two levels of cache, level-one (L1) and level-two (L2) to efficiently transfer the data from/to off-chip memory. When the required data or program code is not present in the L1 cache, it results in L1P and L1D cache misses respectively. Similarly, L2 cache misses can also occur. Cache misses result in stalling of the CPU and can result in more processing cycles. Such overhead can be a key factor in waveforms, e.g. 802.11n with tough constraints, e.g. low latency. Therefore, the mapping process should consider the factors that introduce such overheads and their effects on the whole system performance.

It is obvious from our above discussions that a huge design space for exploration in spatial and temporal mapping exists in the Nucleus concept. The necessity for selecting an appropriate Flavor, after considering other influencing factors, advocates the need for a careful analysis while mapping. Such an analysis can be done efficiently with a tool. A mapping-tool can provide assistance to a WF-developer for identifying an ideal mapping. However, the mapping-tool should be provided with sufficient information in order to consider all the above influencing factors during mapping.

3. TOOL ASSISTED MAPPING AND EVALUATION

Two main requirements of a mapping-tool are: 1) it should identify good, if not optimum, candidate(s) for implementing a WF and 2) it should identify the candidate(s) within a reasonable time. The proposed mapping and evaluation methodology has taken the above requirements into consideration, in addition to the other WF-performance influencing aspects. A two-level evaluation is proposed for fast convergence to WF-implementations (section 3.7).

An important step towards a mapping-tool is to identify the information-sources for the tool. Four key information-sources have been identified. The details that need to be captured from these sources are presented in a structured manner. The same formal structure can be maintained while giving input to the actual tool, e.g. using extensible markup language (XML). To explain the information content, definitions and examples are given at the appropriate sections. Note that a comprehensive list of parameters and properties are beyond the scope of this paper. More details on parameters and properties can be found in [7].

3.1. Overview

Figure 3 gives an overview of the information-sources for mapping process in the Nucleus concept. The goal of the mapping process is to identify a **waveform implementation,** which represents a WF including hardware and software that performs in accordance with the WF-specification. Following are the four key information-sources that are required to reach the mapping-goal:

- A library that consists of Nuclei, **Nucleus library**
- **Waveform description** refers to the set of Nuclei and non-Nuclei connected in a particular fashion to implement the specification of a WF. A WF-description contains constraints that should be met by the WF-implementation (see Figure 3).
- A **Flavor library** is a library that consists of all the Flavors from vendors for a particular HW platform.
- A **HW platform model** is an abstract representation of a HW platform.

A BSP, as shown in Figure 3, consists of a Flavor-Library from vendors for a given HW platform and the HW platform model. We believe that mapping and evaluation will be a process with multiple iterations rather than a single iteration. More details on the information-sources are presented in the following sections. Note that the presented information needs to be provided by the sources to the mapping-tool.

3.2. Nucleus Library

The Nucleus library provides the following information to the mapping-tool. As illustrated in Figure 4, each Nucleus in the Nucleus library is accompanied by a Nucleus model.

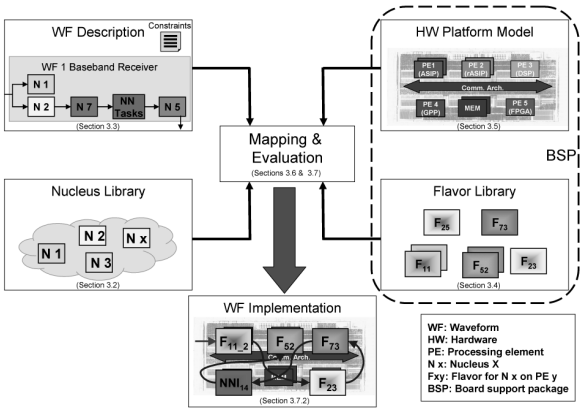

Figure 3. Goal of mapping and evaluation

A **Nucleus model** is an abstract representation of a Nucleus. Some of the information provided by a Nucleus model, like Nucleus parameters, reflects the flexibility of a Nucleus. A Nucleus-model provides the following information:

1) **Name and Description:** Details about the functionality that is implemented by a Nucleus.
Example: Fast Fourier Transform (FFT). FFT is used for efficiently computing a discrete Fourier transform.

2) **Inputs and Outputs:** The input and output data ports of a Nucleus. The characteristics of the ports, e.g. data-width, are provided by the Nucleus parameters.
Example: FFT_in and FFT_out are the input port and output port respectively.

3) Values (or range of values) for **Nucleus Parameters:** A Nucleus parameter is a parameter that belongs to a Nucleus which is used for configuring a Nucleus in accordance with the WF-specification. A Nucleus supports particular value/range-of-values of parameters. Some parameters might have a huge influence on the Nucleus properties. Both Nucleus and Flavor might share some parameters (Figure 4).
Example: Some key parameters are input data-width, input scaling factor, internal scaling schedule and processing mode. The value for input data-width can be 8 to 34 bits.

4) Values (or range of values) for **Nucleus Properties:** A Nucleus property is a property that is used for evaluating the performance of a Nucleus with respect to WF-constraints. Properties are used by the mapping-tool for evaluation and constraint checks. As shown in Figure 4, both Nucleus and Flavor might share some properties.
Example: Some of the performance related properties of a Nucleus are BER, latency, throughput, area, energy and memory. The value for latency can be 16 microseconds.

While the Nucleus-parameters are provided by a Nucleus, the WF-description provides values (or range of values) for the Nucleus-parameters in accordance with the WF-requirements. For example, if the number of FFT-points is a parameter provided by the FFT Nucleus, the WF-

description can provide a value of 512 to the parameter, FFT-points. A value or range of values for the Nucleus properties is provided by the property-equations from Flavors (Figure 4). The influences of some of the properties on parameters of a Nucleus have been investigated in [7].

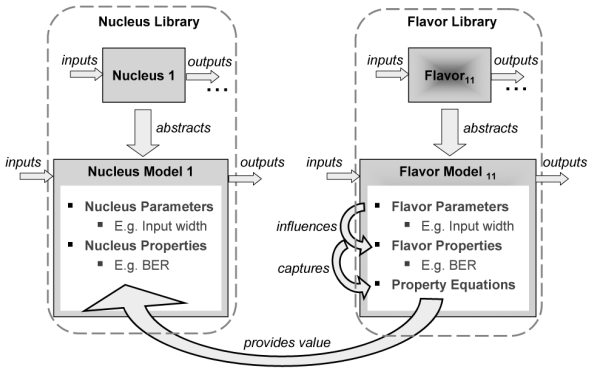

Figure 4: Relation: Nucleus Model and Flavor Model

3.3. Waveform Description

The following information is provided by the waveform description to the mapping-tool:

1) **Nuclei and non-Nuclei:** From the WF-description, Nuclei can be identified and treated separately during the mapping process (using BSP). In general, non-Nuclei can be mapped onto a general purpose processor (GPP).

2) **Data and control flow:** A WF-description provides critical information, e.g. the order in which data and control flows, needed for spatial and temporal mapping.

3) **Constraints:** Constraints of a WF come from two sources: 1) the WF-specification and 2) WF-developer. Constraints like BER, latency and throughput come from the WF-specification while area, energy and memory constraints might come from a WF-developer.

4) **Critical paths:** Usually, a WF-constraint spans few components of a WF, in the receiver or both transmitter and receiver. Many such constraints result in multiple critical paths in a WF involving same/different kernels.

3.4. Flavor Library

A Flavor library, which is part of a BSP, consists of Flavors accompanied by their respective models. A **Flavor model** is an abstract representation of a Flavor. It furnishes the following information to the mapping-tool:

1) **Name and Description:** Details about the implemented Nucleus will be part of the description.
Example: DSP_fft16x32 is an optimized assembly code from TI for the FFT-Nucleus on TMS320C64x [9].

2) **Inputs and Outputs:** The input and output data ports of a Flavor. The characteristics of the ports, e.g. input data-width, are provided by the Flavor parameters.
Example: x and y are the input and output data vectors respectively, whose vector size is the same as FFT-points.

3) Values for **Flavor-parameters:** A Flavor parameter is a parameter that belongs to a Flavor which is used by the mapping-tool for configuring, in accordance with the Nucleus configuration to meet the WF-specification. The flexibility of a Flavor is determined by the supported value/range-of-values of parameters. In general, a Flavor may further restrict the flexibility of a Nucleus to gain efficiency. Apart from the Nucleus parameters, a Flavor may have additional parameters.
Example: Values for input data-width can be 16-, 32-bits.

4) Values for **Flavor-properties:** A **Flavor property** is a property that is used for evaluating the performance of a Flavor with respect to the WF-constraints (Figure 4). Properties are utilized by the mapping-tool for doing evaluation and constraint checks. Flavors might provide trade-offs in terms of properties.
Example: Some examples of the Flavor related properties are BER, latency, energy, etc. As an example, the value for latency can be 16 microseconds.

5) Values for **Flavor-constraints:** A Flavor constraint is a requirement that needs to be fulfilled in order to use a Flavor. Flavor-constraints might have a huge impact on properties.
Example: Input data, if complex, has to be given in interleaved format.

6) **Property Equation:** An equation which gives directly/in-directly the influence of a Flavor on properties in terms of parameters. Most likely, a set of equations is needed for a Flavor, where each equation targets one property in terms of parameters. Property equations are used by the mapping-tool to perform high-level evaluation.
Example: The equation from [9] for DSP_fft16x32 Flavor to calculate the cycles needed for processing a frame is:
$(13 * nx/8 + 24) * \text{ceil}[\log_4(nx) - 1] + (nx + 8) * 1.5 + 27$
where nx is the FFT size.

7) **Simulation Model:** A model that provides output data as a result of processing based on the characteristics of a Flavor on the input data. Models might be available in different accuracy levels.
Example: Some of the accuracy levels are bit-accurate, cycle-accurate, etc.

8) **Interface Definition:** The order how parameters are passed, how input is given to the Flavor and how the results are returned from the Flavor.
Example: void DSP_fft16x32 (const short * restrict w, int nx, int * restrict x, int * restrict y)
where nx is the length of FFT in complex samples, x, y, w are the pointers to input data, output data and FFT-coefficients respectively [9].

Some of the parameters that are "hardcoded" in the Flavor must be available to the mapping-tool, as well.

3.5. Hardware Platform Model

A HW platform model is an abstract representation of a HW platform. Note that more details on the HW platform model can be found in our previous work [10]. A HW platform model provides the following information:

1) **PEs:** Name, type and clock frequency/ranges. Some examples for types of PE are: GPP, DSP, FPGA, etc.

2) **Communication architecture (CA):** Connection links, connected PEs, throughput and latency of connections.

3) **Memories:** Connected PEs, size, type, throughput and latency.

4) **Topology:** Connection of PEs, CA, memories and other peripherals.

3.6. Mapping

We believe that apart from the Flavors, the following factors heavily influence the performance of a WF-implementation.

1) **Data Distribution:** In general, the Nuclei and non-Nuclei kernels need memory for storing internal data like coefficients. The input and output data that is operated by the kernels have to be stored, as well. The location/ distribution of data among the shared resources can make a huge difference in throughput and latency of the system.

2) **Communication:** Kernels need to communicate with each other. Several communication links might exist between the PEs, e.g. a bus or direct links with different bandwidth and latency. The communication link should be selected such that it meets the required bandwidth and latency requirements of the kernels.

3) **Synchronization:** The shared resources, like memory, should be synchronized to avoid resource conflicts. The choice of synchronization method will have a considerable impact on the WF-constraints, e.g. latency.

Apart from identifying a Flavor for a Nucleus, the term "mapping" can be applied to each of the above mentioned factors. For example, "mapping" includes mapping the required communication from the WF-description onto a physical link in the HW platform.

It is obvious that the spatial and temporal mapping of the WF provides a huge design space and highly influences its implementation. Therefore, mapping decisions have to be evaluated against the constraints of a WF. It is important to capture the effects of such overhead and to consider them while mapping. We believe that these effects can be captured in the property equations. For example, an equation for capturing the effects of data distribution is given in [8].

3.7. Evaluation

As illustrated in Figure 5, we propose to evaluate the mapping decisions in two levels. Initially, a high level evaluation is performed at the Nucleus-level (also Flavor-level) based, predominantly, on the property equations. Evaluation at high abstraction-level is fast, but not accurate. Therefore, the results of high level evaluation are accurately evaluated by using simulation models in the low-level evaluation. Two levels in evaluation combine the advantages of abstraction level vs. evaluation speed. Key metrics that can be used for evaluation are listed in [3].

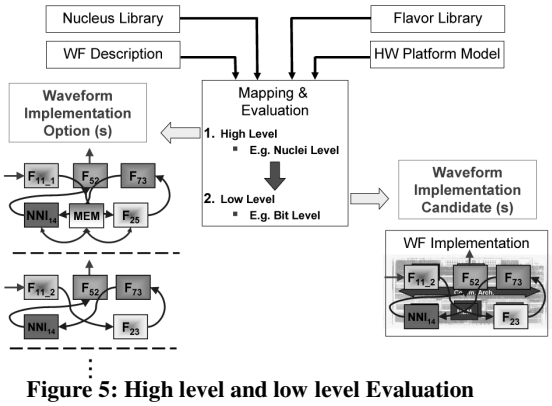

Figure 5: High level and low level Evaluation

3.7.1. High Level Evaluation

An evaluation is done using the Flavor-model, the WF-description, the Nucleus-library and the HW-model to perform the following functions (see Figure 6):

1) Select an appropriate Flavor for a Nucleus that matches the Nucleus parameters.

Example: The value for the number of FFT-points, e.g. 512, must match with the value that is supported by a Flavor.

2) Select the neighbouring Flavors whose parameters relating to the inputs and outputs are in conformance.

Example: Input width should match the output width. More details on this example is given in section 3.8.

3) Select the Flavors which fulfil the WF-constraints using the property equations.

Example: Combined latency of Flavors (and NNI) in a critical path should meet the latency constraint of that path.

4) Select the Flavors whose constraints can be met with minimal (negative) effect on properties.

Example: If the output of a Flavor is in interleaved format, it is beneficial to select a neighbouring Flavor which accepts the input in the same interleaved format.

Note that property equations are not, always, enough for doing high-level evaluation. As an example, for deriving the latency caused by a Flavor (section 3.4, item 6), apart from the processing cycles in the property equation, the clock frequency from the HW platform model is also needed.

Waveform Implementation Option (WIO) As depicted in Figure 5, the outputs of the high-level evaluation are the WIOs. A WIO is an option, consisting of a set of Flavors connected in a particular fashion, to implement the WF that might fulfil the constraints of the WF. There can be several WIOs for one HW platform and each WIO needs low-level evaluation.

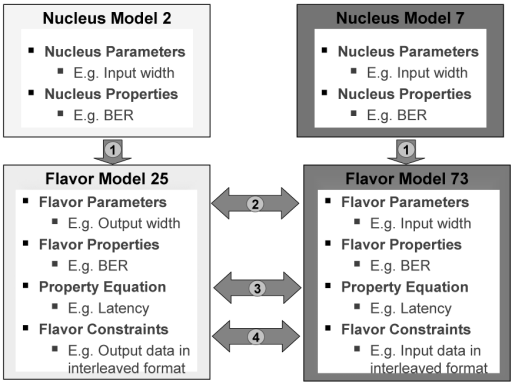

Figure 6: High level Evaluation

3.7.2. Low Level Evaluation

Low level evaluation represents a detailed evaluation on the WIOs coming from the high-level evaluation. Low-level evaluation uses models that are closer to the real HW in order to check the constraints with a higher accuracy.

Example: Instruction or cycle accurate simulation.

Waveform Implementation Candidate (WIC) As shown in Figure 5, the low-level evaluation identifies the WICs out of the WIOs. A WIC is a candidate, consisting of a set of Flavors connected in a particular fashion, to implement the WF that fulfils all the constraints of the WF. A rich Flavor library might lead to several WICs, exhibiting trade-offs in properties. The output of the tool based mapping, with the WIOs as input, is/are the WIC(s).

A two-level evaluation is clearly advantageous for the rapid exploration and mapping of the WF-description. Note that a WF-implementation is one of the WICs. Mapping and evaluation will, most likely, be an iterative process.

3.8. Glue Code Generation

Flavor-constraints, from a Flavor in a WIC, might introduce differences in the data format to its neighboring Flavor. In such cases, additional "glue-code" is needed to adapt the formats. A mapping-tool can be used to generate the glue-code (semi-) automatically, but this requires capturing the differences in data format by the tool. For example, when the output of a Flavor whose data-width is 18-bits is given as input to another Flavor which can support only an input data-width of 16-bits, glue-code is needed to remove the extra 2 least-significant-bits. Note that the impact of removing the bits on properties, e.g. BER, has to be considered.

4. CONCLUSIONS AND OUTLOOK

A methodology of tool-assisted mapping and evaluation for the Nucleus concept has been proposed in this paper. Four key sources of information for the mapping-tool have been identified. The information that needs to be captured from each of the sources has been presented with definitions and examples. Other contributing factors that influence the performance of a WF-implementation have been investigated. The need for considering these factors in mapping has been highlighted with an example. A two-level evaluation has been proposed for a rapid exploration and for an ideal mapping. First, a high-level evaluation, using property equations is done. Following that, a low-level evaluation using simulation models is performed to evaluate accurately.

Future work will involve building tools for supporting the Nucleus concept. Emphasis will be given on (semi-) automatic generation of glue-code for Flavors.

5. ACKNOWLEDGEMENT

The authors would like to thank Weihua Sheng, Stefan Schuermans, Martin Witte, and David Kammler of the ISS institute for inspiring discussions and contributions.

6. REFERENCES

[1] Texas Instruments, "Wimax TI Library," May 2009. http://www.ti.com/corp/docs/landing/ wimax/index.htm

[2] Lyrtech, "Developing a GSM Modem on a DSP/FPGA Architecture," May 2009. www.lyrtech.com/publications/ Article GSM Modem DSP.pdf

[3] V. Ramakrishnan, et al., "Efficient and Portable SDR Waveform Development: The Nucleus Concept," to appear in IEEE *Military Communications Conference (MILCOM 2009)*, http://arxiv.org/abs/0906.3313

[4] M. Wesseling, "Modeling Language For Software Defined Radio Applications," in Proceeding of the SDR 05 Technical Conference and Product Exposition.

[5] C. Grassmann, et al., "Mapping Waveforms to Mobile Parallel Processor Architectures," in Proceeding of the SDR 05 Technical Conference and Product Exposition.

[6] K. Maier, "Mapping waveforms to systems: What would a wideband networking waveform system require?," in Military Embedded Systems Magazine, October 2005.

[7] V. Ramakrishnan et al., "Efficient Implementations From Libraries: Analyzing the Influence of Configuration Parameters on Key Performance Properties," in IEEE *Personal, Indoor and Mobile Radio Communications Symposium (PIMRC '09)*, Tokyo, 2009.

[8] Texas Instruments, "Signal Processing Examples Using TMS320C64x DSPLIB," September 2003. [Online]. Available: http://focus.ti.com/lit/an/spra884a/spra884a.pdf

[9] Texas Instruments, "TMS320C64x DSP Library Programmers Reference," October 2003. http://focus.ti.com/lit/ug/ spru565b/spru565b.pdf

[10] T. Kempf et al., *"An SDR Implementation Concept based on Waveform Description"*. in FREQUENZ: Journal of RF-Engineering and Telecommunications (9-10), Berlin, 2006.