A Case Study in Performance Estimation and Analysis for SDR: UMTS Data-Link Layer

Alena Simalatsar and Roberto Passerone
August 2007

Technical Report Number: DIT-07-057

A Case Study in Performance Estimation and Analysis for SDR: UMTS Data-Link Layer

Alena Simalatsar and Roberto Passerone
Dipartimento di Informatica e Telecomunicazioni
University of Trento
{simalats, roby}@dit.unitn.it

Abstract

The evolution of technology in the wireless industry has brought a proliferation of different communication protocols and standards. To support a wider range of services and to sustain technical innovation, protocol interoperability is essential. Software Defined Radio (SDR) is emerging as a key integration technology in this area. Defining the appropriate system architecture for SDR is, however, complex, and requires extensive analysis and performance evaluation. In this paper, we consider UMTS as a case-study of a mobile standard that utilizes several high-layer protocols to enable multiple services delivery for mobile users. We develop a SystemC functional model for a subset of the Data Link Layer for UMTS, and analyze its performance as software running on several architectures. The results are a first step in understanding the system requirements to combine UMTS with other wireless communication protocols, such as WiFi and WiMax.

1. Introduction

The construction of a distributed communication infrastructure, and the use of highly connected embedded systems, makes it possible today to realize new and innovative applications and services, often context-aware, that can leverage the mobility afforded by wireless connectivity [5]. While these services are interesting in their own right, their integration could potentially lead to a much greater added value. However, the wide variety of different communication requirements, together with the evolution of technology, have resulted in a proliferation of mostly incompatible and often competing communication standards, that make integration hard and expensive. The result is that users are forced to carry multiple devices, and manually synchronize and duplicate information, a potentially onerous and error-prone task.

One solution to the integration problem is the use of a Software Defined Radio (SDR) [19]. In a pure SDR, all the required processing, i.e., tuning, modulation and demodulation and the handling of the higher levels of the protocols, is done in the digital domain by software running on a general purpose processor, while a slim RF front-end remains in the analog domain. Because the task is highly demanding in computation power, most platforms include dedicated hardware components that can be configured and rewired by software to adapt to the required functionality. These architectures are very flexible and allow one to switch between different communication standards by simply uploading a new version of the software in the device, and potentially support several protocols at the same time. Not only does this make integration simpler, but it also allows the devices to be updated to support new standards in a way that is mostly transparent to the user.

The implementation of a SDR requires the development of highly optimized computing platforms. However, the tight real-time requirements of communication systems, together with constraints on cost, physical size and performance of the devices, results in increased design complexity and system heterogeneity that yield a large design space. Tools based on Register Transfer Level (RTL) and evaluation boards are too detailed for an effective exploration of system design alternatives, and are typically biased towards specific implementation styles. In addition, the lack of abstraction makes the design, as well as the validation process, difficult.

In this paper, we propose to raise the level of abstraction for the design of SDR platforms using the Platform-Based Design (PBD) [10, 20] methodology. PBD is based on the formation of different layers, called platforms, which represent different levels of the design abstraction. Each platform is a well separated library of computational and communication components. Platforms at higher levels abstract the details of lower level platforms, and can be used for fast architectural exploration and performance estimation. This is essential for quickly converging toward a platform that is not only optimized for the desired functionality, but can also support its future extensions. For the methodology to work, each domain and application area requires the development of distinct libraries of components, abstraction layers, and thus architectural, performance and functional models. The effectiveness and the acceptance of the PBD methodology is therefore tied to the availability of these models and to the development of use cases that can prove their accuracy.

In particular, in this paper, we strengthen the PBD methodology by developing, using existing tools [13, 22], appropriate functional and architectural models for SDR. These model are used in a case study to explore the impact of higher protocol layers on the utilization of a general purpose CPU, and derive performance metrics that can be used to optimize the parameters of an architecture, based on the forseen mix of concurrent protocols. We show that PBD is particularly well suited for the design of SDR, since the requirements in terms of performance and adaptability are high, and the definition of the best architecture for this kind of systems is hard.

This paper is organized as follows. After discussing some related work in Section 1.1, we give an overview of our methodology and of the domain of application in Section 2. Section 3 discusses in detail the functional and architectural model. Finally, simulations and results are discussed in Section 4.

1.1. Related Work

The literature on design space exploration is vast. In this section, we will limit our exposition to the work that appears to be closer to our defined area of application.

In the area of enabling technologies for reconfigurable wireless terminals, the XiRisc group of the University of Bologna in collaboration with STMicroelectronics [23] focuses on the development of next generation mobile technologies based on reconfigurable terminals [4,7]. The architecture for reconfigurable terminals presented by this group is able to support some important mobile technologies (GSM, WCDMA, UMTS, WLAN and Bluetooth) and allows software driven switching between these technologies. However, the aim of this work is to provide an application-specific architecture, rather than a design methodology. Nevertheless, this architecture may be a potential target for our design space exploration.

Boni et al. have proposed a technique for the efficient implementation of support vector machine (SVM) algorithms on dynamically reconfigurable systems [8]. Specifically, they have proposed a reconfigurable system to solve the inverse modeling problem for telecommunication applications by the use of SVMs. While they do not address the problem of design exploration in general, the solutions presented in their work for the dynamic mapping of algorithms onto reconfigurable FPGA-based architectures can be used in our future work.

Bonivento et al. are also working on several projects related to design space exploration, including the area of wireless communication in the form of wireless sensor networks [9]. In particular, they are developing a new methodology for the design of Wireless Sensor Networks (WSN) applying PBD principles, which is well correlated with the nature of our research work. However, they focus on high level operating system services, while we will focus on lower level hardware architectures.

Of particular interest for our research is the work of Densmore et al. which is focused mainly on system level architecture modeling based on the PBD methodology. In particular, they are working on performance analysis of FPGA-based architectures [12]. While we can take advantage of these results in our architecture models, our research work is not just focused on such kind of architectures and we will need to consider other alternatives for architecture implementations.

Davare et al. propose a methodology for enabling automated synthesis within PBD. Their work is mainly focused on concurrent embedded design for multimedia applications [11]. In this context, they propose functional optimization using design space exploration for multimedia applications by applying the process of mapping of these applications onto FPGA-based architectures [14]. To do this, they use the Metropolis design framework [6]. Our approach will be similar. What distinguishes our research work from theirs is that we will do design space exploration for architectures that can support concurrent protocols instead of multimedia applications.

The research work of Meyerowitz et al. is the most related to our project [18]. Their work focuses on high level modeling of embedded micro-architectures retargetable for different instruction sets. The modeling of micro-architectural performance is done within the Metropolis framework. Recent presentations suggest the use of this technique for Software Defined Radio, an approach similar to ours (in SystemC) for architecture design space exploration [17]. The lack of specific publications in this area from the authors, however, makes a comparison difficult at this point. Nevertheless, the availability of future results will be instrumental to validate our architecture exploration methodology and models.

2. Overview

A Software Defined Radio (SDR) is a technology in which both modulation and demodulation are performed in software or using a programmable device [19]. The major advantages of an SDR is its flexibility and ease of adaptation, since the radio function can easily be changed. Programmability also promises economy of scale for manufacturers, who can rely on standard platforms reused across different domains of applications. A typical architecture for an SDR platform is shown in Figure 4.

The design of reusable platforms is, however, hard. The designer must carefully consider several different scenarios in the choice of the architectural components, and in the way they are connected. For instance, Shono et al. report that the arrangement of CPUs, DSPs and FPGAs seriously influences system performance, and that the software assignment to each processor is difficult [21].

To define an effective system design flow for SDR, we apply the PBD paradigm [20], by evaluating different architectures against the specification constraints, and by mapping the desired functionality on the elements of the platform. The main objective of the design exploration process for each particular architecture is to define what and how many protocols can be supported by the platform. To facilitate this process, it is important to separate the architecture specification from the functionality. This way, changes in functionality (or in the architecture) will not cause the redesign of the entire system, and vice versa. In addition, this allows us to model function and architecture at two different levels of abstraction, and enable fast annotated functional simulation to quickly provide performance metrics for a variety of design choices.

Our particular implementation of the platform-based design methodology follows the steps presented in Figure 1. We first build an abstract SystemC model of the functionality, with no notion of time (the Untimed level in the figure), which is used to verify its correctness and to study concurrency issues. SystemC was chosen because it supports different models of computation and allows the design of heterogeneous systems. In addition, it provides the possibility to refine high level specification models (both hardware and

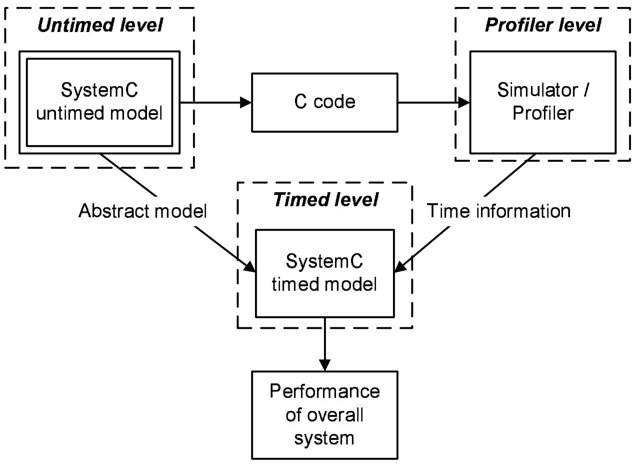

Figure 1. Design exploration methodology

software) into low level implementations and to build executable models. Finally, because SystemC is based on a standard language (C++), it is easy to share models with the other members of the research group, such as developers of applications, UI and baseband.

In order to create a SystemC performance model for a particular architecture we need to know the performance of each functional block executed on this architecture. To do so, we extract the C code from our SystemC model and run simulations on several emulated ARM processors using the Keil ARM Development Tool [15]. We use Keil's embedded profiler to obtain information about the execution time of each function run on each emulated microcontroller. Keil implements a lower level of abstraction (the Profiler level), which we use to extract the relevant performance data. We then use the timing information to build a complete SystemC timed model (the Timed level), which we use to determine the performance of the overall system, and to compute the utilization of the microcontroller for data transmission with several bit rates.

This scheme has several advantages over the use of the profiler alone, which by itself can provide the system performance. First, the simulation run in evaluation mode are very time consuming and depend on the complexity of the microcontroller architecture. In contrast, SystemC simulations are relatively fast and independent of the microcontroller architecture (see Section 4). Second, SystemC is more flexible and makes it easier to partition the functionality onto different processor cores, and to combine their performance. This is essential as platforms evolve to include more processing elements. This trend also requires the exploration of different concurrency models, from dataflow to synchronized execution, which is natively supported by SystemC but typically not by architecture profilers.

In this paper we focus on the UMTS communication standard [2]. At the top level, the network architecture is divided into a User Equipment Domain and an Infrastructure Domain, which communicate through the radio interface. We focus on the User Equipment Domain, which is of greater interest to mobile devices which are subject to more stringent implementation constraints.

The protocol stack for the user domain is shown on Figure 2, and is divided into three different layers, corresponding to the Physical (PHY), the Data Link (DLL) and the Network Layer [3]. In addition to the usual addressing functions, the Network Layer controls the operations of the Data Link and of the Physical Layer by responding to changes in the transmission parameters. The Data Link Layer performs general packet forming and quality of service support. It includes a Radio Link Control (RLC) and a Medium Access Control (MAC) sublayer. The RLC communicates with the MAC through different *logical* channels, to distinguish between user data, signaling and control data. Depending on

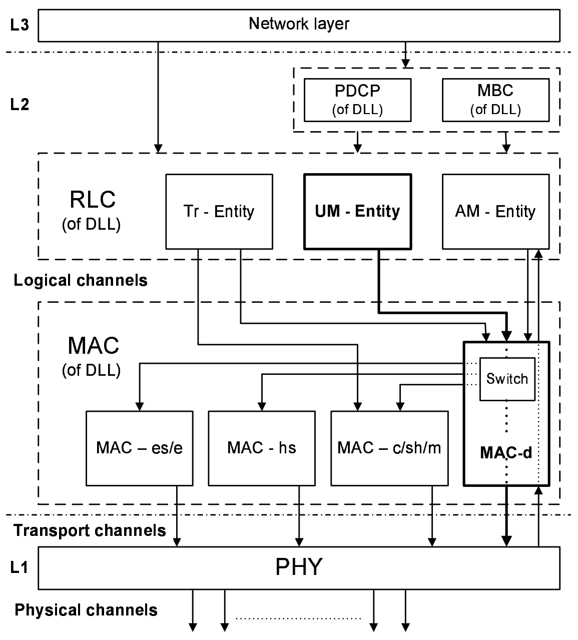

Figure 2. Layer diagram for the User Equipment Domain of the UMTS protocol

the required quality of service, the MAC layer maps the logical channels into a set of *transport* channels, which are then passed onto the Physical Layer. Finally, the Physical Layer handles lower level coding and modulation, and communicates with the radio interface through a series of *physical* channels, each optimized to different time and coding requirements.

The architecture of the protocol stack is very complex due to the high number of different logical and transport channels. In this work we focus on a subset of the functionality, described in the next section together with the architecture chosen for performance estimation.

3. Functional and Architecture Model

In this section, we describe in more detail the functionality of UMTS that we modeled, and the class of architectures that we have considered as target for the performance analysis. Detail results are presented later, in Section 4.

3.1. Functional Model

The subset of the protocol stack that we have modeled is the section highlighted on Figure 2, and corresponds to the bidirectional Dedicated Channel that, in our case, is limited to point-to-point uplink user data transmission. The RLC is divided into three separate entities for Transparent (Tr), Unacknowledged (UM) and Acknowledged (AM) transmission modes. We have limited our analysis to the Unacknowledged mode, corresponding to the UM-Entity, since this is a superset of the Transparent mode, and can be used to a certain degree to estimate the performance of the Acknowledged mode, which would otherwise require the downlink model. Thus, the results of performance analysis of the UM-Entity can allow us to make approximate estimations of the performance of the complete RLC layer.

Likewise, the MAC sublayer is divided into different entities that handle the mapping between the logical and the transport channels. Of these, we model the MAC-d entity which is the only one involved with the baseline (not enhanced) Dedicated Channel. The other blocks, which were introduced in more recent versions of the standard, are instead required for high speed and quality of service support.

Our SystemC untimed functional model is shown on Figure 3 and is composed of seven modules, or actors, of a dataflow process network: four are related to the UM RLC entity (i.e., Transmission Buffer, Segmentation/Concatenation, RLC Header Add and Ciphering) and three to MAC-d (i.e., Transport Channel Type switch, C/T MUX and Transport Format Combination (TFC) selection).[1]

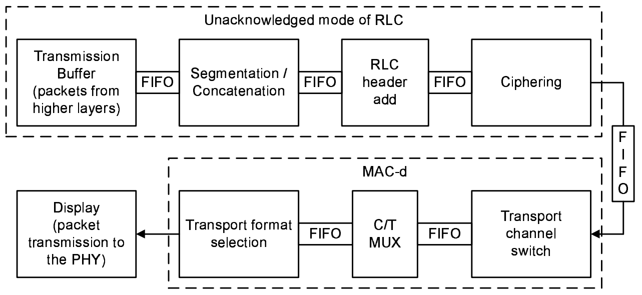
Unacknowledged mode of RLC
Transmission Buffer (packets from higher layers)
FIFO
Segmentation / Concatenation
FIFO
RLC header add
FIFO
Ciphering
FIFO
MAC-d
Display (packet transmission to the PHY)
Transport format selection
FIFO
C/T MUX
FIFO
Transport channel switch

Figure 3. Block diagram of the functional model

Each block signals to the next the availability of a packet to be processed, by depositing it into a FIFO queue with blocking read and blocking write. The scheduling of the network is then left to the simulator.

SystemC timed functional model has the same composition of block. We assume that all the blocks are executed on the same processor (ARM processor) that means than while one of the blocks is executed others are waiting for it to be finished. Execution of each process takes a particular time which we add to the SystemC simulation time while processing each of the blocks and we assume that this time is already known from running the functionality on an emulator of explored processor. This way, the beginning and the

[1] MAC-d typically includes also a ciphering module, but since encryption is already done at the RLC layer, this module need not be implemented.

end of each process execution has its own time stamps.

This model can be extended by including more entities of the RLC and the MAC sublayers to obtain more accurate results. Our future work includes also extensions to the network layer and higher level application such as MPEG decoding and the User Interface (UI).

3.2. Architecture Model

To design an optimal architecture we need to decide what elements should be available on the platform to achieve the best trade-off between the metrics of interest. These elements include general-purpose processors, Digital Signal Processors (DSP), Field Programming Gate Arrays (FPGAs), or their mix. This step also includes identifying the kind of processors to be used (and their performance), as well as their number and general interconnection topology. To begin with, we surveyed several architectures proposed by the industry for SDR, and finally decided to take the Small Form Factor SDR (SFF SDR) Development Platform from Lyrtech [16] as our baseline model. We can then use the architecture of this platform as a starting point and modify it for the design space exploration of an SDR.

The SFF SDR platform consists of three separate modules, the Radio Frequency (RF), Data Conversion and Baseband Processing modules, combined together, and is shown in Figure 4. Each of these modules can be replaced in order to satisfy the requirements of the target product. The Baseband Processing module works in the digital domain, and is the main focus of our studies. This module employs a TMS320DM6446 system-on-chip (SoC) from Texas Instruments and a Virtex-4 SX35 FPGA from Xilinx for modulation. The SoC consists of one C64x+^{TM} DSP and one ARM926EJ-S^{TM} general-purpose processor. The DSP is typically used for processing the baseband, while the general-purpose processor is reserved for the upper layers of the communication protocols and other higher level applications.

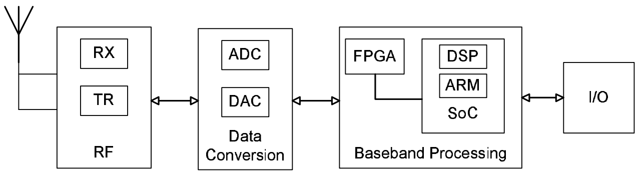

Figure 4. Block diagram of the SFF SDR

In this paper, we restrict our attention to the ARM family of processors, since the functionality that we are testing is limited to a subset of the UMTS DLL layer. We consider less performing processors than the one available on the SFF SDR platform (which we only take as a template), because of the limitations imposed by the profiler that we have used. Our future work includes extending the performance analysis to higher performance processors (such as the one used on the SFF SDR), and integrating the baseband processing, currently under development, which we assume is realized partly on the FPGA, and partly on the DSP.

4. Simulations and Results

This section is devoted to presenting the results of the performance analysis for the RLC and the MAC sublayers. We use three kinds of metrics to characterize the performance of different architectures, to determine the distribution of resources within the protocol function, and to measure the efficiency of the performance analysis itself.

Our first results, depicted in Figure 5, show the percentage of utilization (load) of the general-purpose processor under different transfer rates, and for different architectures. We have analyzed five different ARM microcontrollers that include the STR912FW44, STR750FV2 and STR736FV1 from STMicroelectronics, and the LPC2119 and LPC2194 from NXP. These have been analyzed at their maximum clock frequency (shown in Figure 5 under the name of the architecture), with the exception of the STR912FW44 and the STR750FV2. The first (STR912FW44) was simulated at 60 MHz, instead of 96 MHz, for an easier direct comparison with the NXP microcontrollers. The second (STR750FV2) was simulated at 25 MHz, instead of its maximum 60 MHz, to obtain roughly the same performance results as the STR736FV1, and therefore show the savings in clock frequency.

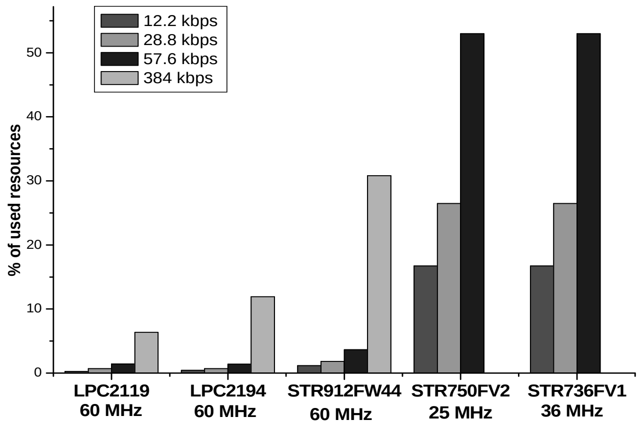

Figure 5. Architecture performances

For each microcontroller, we have computed the load for the four most common transmission data rates, i.e., 12.2 kbps for voice communication, 28.8 kbps and 57.6 kbps used by modems and faxes, and 384 kbps for high speed data transmission. For each data rate we use a fixed, though different, packet size for the Transport channel. For instance, we use a Transport block size of 576 bits for modems and faxes, and of 336 bits for high speed data transmission [1]. Each packet has a 16-bit overhead for RLC and MAC headers. In addition to that, information data delivered by the Dedicated Transport Channels (DTCHs) are accompanied by control packets delivered through the Dedicated Control Channels (DCCHs). In our simulations, we have assumed that each DTCH packet is accompanied by one DCCH packet. For a bit rate of x kbps, the load is computed as the ratio between the actual time it takes to transmit x bits, and one second, the maximum time allowed for the transmission. Formally, we have

$$L = \frac{(T_{dp} + T_{cp}) \cdot x/8}{P_{dp} + P_{cp}} \cdot \frac{1}{1s} \cdot 100\%,$$

where T_{dp} and T_{cp} is the time to transmit a data and a control packet, and P_{dp} and P_{cp} is the size, in bytes, of data and control packets, respectively. For the control packet we have assumed a constant size of 100 bits.

The results of the analysis presented in Figure 5 show considerable variability, both across architectures, and, as expected, for different data rates. This analysis gives us a measure of the residual computing power available to the rest of the protocol, to potential other protocols running concurrently, and to higher level applications, which is an essential information for the correct architecture choice and sizing of the system. In the case of the STR736FV1 and STR750FV2, the load for 384 kbps exceeds 100%, and is therefore not shown on the figure.

A second class of results is devoted to the analysis of the resources required by each of the functions of the model shown in Figure 6. These results are for the STR912FW44

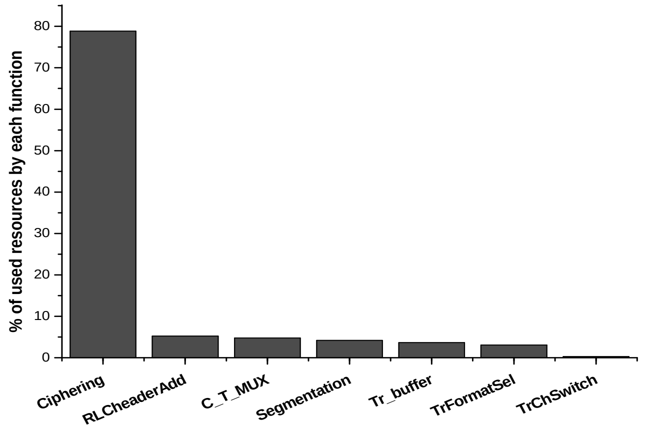

Figure 6. Resource distribution by function

ARM9 microcontroller from STMicroelectronics and the packet size of 70 bytes, which we have taken as representative, since the analysis for the other architectures and packet sizes are qualitatively and quantitatively similar. The functions on Figure 6 are sorted by the amount of resources that they require. As we can see from the graph the Ciphering function is in general the most resource consuming function. This kind of result is very useful for taking the preliminary decision about resource distribution between the functions.

Our last group of results is concerned with a measure of the efficiency of the performance analysis. In Table 1, we show, for each architecture, the time in seconds required to simulate the transmission of 7,000 packets, with a packet size of 70 bytes, for both the Keil profiler and SystemC. The simulations were performed on a Pentium^R M processor running at 1.70 GHz, with 512 MB of RAM. For SystemC, we have used the reference implementation version 2.1.v1, available at the SystemC web site [22].

Architecture	**Keil**	**SystemC**	**Speed-up**
LPC2119	28.1	4.2	6.7
LPC2194	25.8	4.2	6.1
STR912FW44	25.5	4.2	6.1
STR750FV2	26.6	4.2	6.3
STR736FV1	25.9	4.2	6.2
Average	26.4	4.2	6.3

Table 1. Efficiency of performance analysis

The performance of the SystemC simulation is independent from the architecture, since it is referred to the same model. In contrast, Keil shows variability depending on the microcontroller. The last column of the table shows the speed-up obtained by using the more abstract SystemC models. On average, the SystemC simulation is more than 6 times faster than the profiler, which justifies the use of the profiler for data gathering, and the use of the abstract simulator for performance analysis. This methodology becomes even more compelling as the complexity of the model increases, and several protocols are run concurrently, since the simulation time will correspondibly increase. The performance of the SystemC simulation can potentially be increased by using an optimized simulator rather than the reference implementation.

5. Conclusions and Future Work

We have presented a case study of a design space exploration problem for a Software Defined Radio. We have selected a subset of the Data Link layer of the UMTS communication standard, and evaluated its performance on several architectures, and for different data rates and packet sizes. Our methodology uses separated functional and architectural models. The functional models run at a high level of abstraction, and can be used for functional verification and fast performance analysis through execution time annotation. The performance model is used to derive accurate performance data, and uses a profiler to simulate the exact behavior of the architecture. The analysis shows considerable variability in performance, both as a function of architecture, and as a function of data rate. This justifies the use of the PBD approach to design this kind of systems, which must be fine tuned to be able to support different standards.

Our future work is focused towards concretizing the PBD approach into a specialized methodology for the design space exploration of communication hardware architectures. This includes automating the process of annotation through the use of dedicated components, able to also determine the share of CPU in the case of multi-threaded execution. In addition, we will develop interaction components for the estimation of communication overhead in multi-component architectures, which are common in SDR to achieve the desired performance. Our future work also includes the extension of the functional model to integrate the baseband processing (on the DSP), higher levels of the protocol, and other protocols of interest such as WiMax, WiFi and GSM.

6. Acknowledgments

The authors would also like to thank Fernando Pianegiani and Fabrizio Stefani for stimulating discussions and suggestions. This work was supported in part by a grant from ArsLogica, s.r.l.

References

[1] 3rd Generation Partnership Project. Channel conding and multiplexing examples. Technical Specification TS 25.944, 3GPP, June 2001.

[2] 3rd Generation Partnership Project. General universal mobile telecommunications system (UMTS) architecture (release 6). Technical Specification TS 23.101, 3GPP, December 2004.

[3] 3rd Generation Partnership Project. Radio interface protocol architecture (release 7). Technical Specification TS 25.301, 3GPP, March 2006.

[4] F. Agnelli, G. Albasini, I. Bietti, A. Gnudi, A. Lacaita, D. Manstretta, R. Rovatti, E. Sacchi, P. Savazzi, F. Svelto, E. Temporiti, S. Vitali, and R. Castello. Wireless multi-standard terminals: system analysis and design of a reconfigurable rf front-end. *IEEE Circuits and Systems magazine, Special Issue on Wireless Reconfigurable Terminals*, 6(1):38 – 59, 2006.

[5] C. Andersson. *GPRS and 3G Wireless Applications*. John Wiley & Sons, April 2001.

[6] F. Balarin, L. Lavagno, C. Passerone, A. L. Sangiovanni-Vincentelli, M. Sgroi, and Y. Watanabe. Modeling and designing heterogeneous systems. In J. Cortadella and A. Yakovlev, editors, *Advances in Concurrency and System Design*. Springer-Verlag, 2002.

[7] A. Baschirotto, F. Campi, R. Castello, G. Cesura, R. Guerrieri, L. Lavagno, A. Lodi, P. Malcovati, and M. Toma. Baseband analog front-end and digital back-end for reconfigurable multi-standard terminals. *IEEE Circuits and Systems magazine, Special Issue on Wireless Reconfigurable Terminals*, 6(1):8 – 28, 2006.

[8] A. Boni, F. Pianegiani, and D. Petri. Inverse modeling with SVM-based dynamically reconfigurable systems. In *Proceedings of the IEEE Instrumentation and Measurement Technology Conference (IMTC)*, pages 2036–2040, Como, Italy, May 2004.

[9] A. Bonivento, L. P. Carloni, and A. L. Sangiovanni-Vincentelli. Platform based design for wireless sensor networks. *MONET*, 11:469–485, 2006.

[10] L. P. Carloni, F. D. Bernardinis, A. L. Sangiovanni-Vincentelli, and M. Sgroi. The art and science of integrated systems design. In *Proceedings of the* 28th *European Solid-State Circuits Conference, ESSCIRC 2002*, Firenze, Italy, September 2002.

[11] J. Chong, A. Davare, and K. Lwin. Concurrent embedded design for multimedia: JPEG encoding on Xilinx FPGA case study. Technical Report UCB/EECS-2006-40, University of California, Berkeley, April 16 2006.

[12] D. Densmore, A. Donlin, and A. L. Sangiovanni-Vincentelli. FPGA architecture characterization for system level performance analysis. In *Proceedings of the Conference on Design, Automation and Test in Europe (DATE06)*, Munich, Germany, March 6–10, 2006.

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

[14] S. Kakita, Y. Watanabe, D. Densmore, A. Davare, and A. L. Sangiovanni-Vincentelli. Functional model exploration for multimedia applications via algebraic operators. In *Proceedings of the* 6th *International Conference on Application of Concurrency to System Design*, Turku, Finland, June 26–30, 2006.

[15] Keil. http://www.keil.com.

[16] SFF SDR development platform. Technical specifications, Lyrtech, February 2007.

[17] T. Meyerowitz, R. Chen, A. L. Sangiovanni-Vincentelli, and J. Harnish. Modeling a heterogeneous multiprocessor for software defined radio. In *Presentations of the CHESS review meeting*, Berkeley, CA, February 2006.

[18] T. C. Meyerowitz and A. L. Sangiovanni-Vincentelli. High level CPU micro-architecture models using Kahn process networks. In *SRC TechCON*, Portland, Oregon, October 24–25 2005.

[19] J. Mitola. The software radio architecture. *IEEE Communication Magazine*, 33(5):26–38, May 1995.

[20] A. L. Sangiovanni-Vincentelli. Defining platform-based design. *EEdesign*, February 2002.

[21] T. Shono, Y. Shirato, H. Shiba, K. Uehara, K. Araki, and M. Umehira. IEEE 802.11 wireless LAN implemented on software defined radio with hybrid programmable architecture. *IEEE Transactions on Wireless Communications*, 4(5):2299–2308, September 2005.

[22] SystemC. http://www.systemc.org.

[23] XiRisc. http://www.micro.deis.unibo.it/ campi/XiRisc.