Simulation framework for early phase exploration of SDR platforms: a case study of platform dimensioning

Martin Trautmann*†, Stylianos Mamagkakis†, Bruno Bougard†, Jeroen Declerck†
Erik Umans†, Antoine Dejonghe†, Liesbet Van der Perre† and Francky Catthoor*†
*Katholieke Universiteit Leuven, Belgium
†Interuniversity Microelectronics Center (IMEC), Leuven, Belgium
trautman,mamagka,bougardb,dclerckj,umans,dejonghe,vdperre,catthoor@imec.be

***Abstract*—Software Defined Radio (SDR) terminals are crucial to enable seamless and transparent inter-working between fourth generation wireless access systems or communication modes. On the longer term, SDRs will be extended to become Cognitive Radios enabling efficient spectrum usage. Future communication modes will have heavy hardware resource requirements and switching between them will introduce dynamism in respect with timing and size of resource requests. In this paper, we propose a modeling framework that enables the simulation of such complex, dynamic hardware/software SDR designs. Thus, we can do an exploration, which can pinpoint the coarse grain platform component requirements for future SDR applications in a very early design phase. Our solution differs from existing ones by combining multiple simulation granularities in a way that is specialized for SDR simulation. Finally, we demonstrate the effectiveness of our approach with a case study for dimensioning the on-chip interconnect of a prospective SDR platform.**

I. Introduction

Both rising performance requirements of embedded applications and manufacturing cost pressures drive the embedded system design industry to integrate an increasing number of hardware components on a single chip. This trend combined with power consumption issues led to the rise of the Multiprocessor System-on-Chip paradigm which renders the design process of hardware and software even more complex. Furthermore, it is widely acknowledged that early design decisions have the most significant impact on the final system performance and power consumption [1].

In the application domain of Software Defined Radio (SDR) and Cognitive Radio, the demand for increased performance is accompanied by an increase in user control and interactivity with the environment. This has a significant impact on the resource requests, which become more dynamic because different wireless protocols (with different range of performance requirements) can be switched at unknown timing moments.

Traditionally, the design and development of these SDR hardware platforms uses worst case estimations of the aforementioned dynamic software requirements in order to decide on hardware resource allocation. However, it has been shown that run-time resource management optimizations can reduce resource requirements of wireless applications significantly without affecting the Quality of Service (QoS) and Experience of the final user [2]. These savings of run-time mechanisms should be taken into account, so that there is no unwanted over-provisioning of resources.

Software developers have trouble providing performance characteristics of their software including the effects of run-time mechanisms, at a time when no source code exists that is ready for optimized mapping to any relevant target platform. In order to take into account the various dependencies between hardware and software development and their refinement efforts, a Hardware-Software co-design simulation framework should be available that can deal with a combination of hardware models, software models, and run-time models. Moreover, it should be feasible to create this in a very early stage of the development process. This combination of properties is not available for existing simulation frameworks (see section II).

In this paper we present a simulation framework designed for HW/SW co-design of SDR systems in an early development stage. Furthermore, we describe a case study where we use the simulator to analyze a run-time scheduling technique and the bandwidth requirements for the on-chip interconnect of a prospective SDR platform. The structure of this paper is as follows. The next section describes related work. In section III we describe our simulation framework. In section IV we describe the case study of dimensioning the interconnect of a prospective SDR platform. Finally, in section V we conclude the paper.

II. Related Work

Component-based embedded systems are realized today by using a multitude of heterogeneous execution and communication models. The hardware and software components are put together by using an environment like Matlab/Simulink, Metropolis [3], or Ptolemy [4]. Additionally, UML has emerged in recent years as a modeling standard for software [5], including also software for embedded systems for which specific UML profiles have been developed (e.g. MARTE for real time embedded systems [6]). In the context of this paper, we aim at modeling embedded systems of the SDR application domain for the purposes of early design phase dimensioning

978-3-9810801-5-5/DATE09 © 2009 EDAA

of the system component requirements and parameters. The aforementioned modeling environments are more rich and able to address more design issues (e.g. verification, model transformations, etc.) for any application domain and in that sense they offer viable extension solutions for our specialized modeling environment. But they are too slow for the fast early exploration goal we target. Moreover, they would require a significantly higher modeling effort for creating system and software models which enable the necessary control options for an encapsulated run-time manager model.

In respect with on-chip interconnect simulation, most of the related work for MPSoC platforms has been focused on Network-on-Chips [7]. WormSim [8] is a NoC simulator which focuses on communication aspects and its flexibility allows such an exploration. Nostrum NoC Simulation Environment [9] is another example of such a flexible simulator which focuses on grid-based, router-driven on-chip communication. MPARM [10] allows both a detailed simulation of computation and communication and speeds up the execution by emulation. Finally, a NoC platform designed using Platform Architect [11] can also be used. Platform Architect is based on SystemC and is used for SoC design with Transaction-Level Modeling (TLM). In the context of this paper, we aim at faster, early design phase exploration and coarser grain dimensioning of the interconnect. The aforementioned simulators can be used for increasing accuracy at later design phases, when the exploration space is narrowed.

III. Simulation Environment for early phase SDR Platform Exploration

The simulation framework presented in this paper is designed for SDR platform exploration in an early design phase. The software modeling used for driving the platform exploration allows to take run-time resource management solutions into account by providing realistic input data during the simulation.

A. Choice of Simulation Accuracy

In an early design phase it is sufficient to use models which are less than cycle accurate such as task graph based software modeling and transaction level hardware modeling, because neither the hardware platform instance is selected nor the software source code is refined to its final form. Tasks representing hundreds of instructions or more are a manageable organization unit which simplifies software modeling but still enables to explore parallelization options for an MPSoC target platform. Parameters of models of future embedded systems are derived from experience with previous implementations through extrapolation. Task graph granularity timing modeling has also been used by previous research like [12].

B. Functionality Simulation in addition to Timing Simulation

Task graph models for simulating the timing of applications running on an MPSoC platform can quickly be generated from software partitioning ideas and execution time estimations. When simulating run-time management techniques, however, a pure timing model is not sufficient. In order to simulate different behavior as a result of different input data or event timing, it is important to simulate the application functionality as well. The functionality has to be simulated to such an extent that relevant input data and input dependent event timing can be provided to the run-time scheduler model during simulation.

Extending a task graph timing model to a fully functional model greatly complicates the modeling effort. Instead of modeling the function for each task in the task graph, we model it on packet level granularity only. This is enough for both providing a run-time scheduler with realistic input data and giving it control over allocations and timing on task graph granularity. To the best of our knowledge, this specific combination of using multiple granularities for function and timing simulation of SDR baseband processing has not been used in research so far.

For exploring SDR platforms we only model baseband processing software detailed enough to schedule it flexibly on platform resources. In order to generate realistic inputs from a higher layer of the protocol stack, we co-simulate baseband processing with a model of the time critical part of the 802.11 MAC layer. The latter includes the Distributed Coordination Function (DCF) which controls timing and back-off such that multiple terminals can transmit on the same channel with minimized risk of collisions. Furthermore, we can simulate multiple protocol stacks in parallel and map their baseband processing task graphs on a single platform model. This allows us to model resource contention for an SDR supporting multiple simultaneously operating standards and to study run-time optimization techniques which deal with this kind of contention.

C. Components

The components that make up the simulation framework are shown in figure 1. The physical layer implementation is based on a simulation server layer that provides an event based simulation kernel and an air-channel model.

The model of the 802.11n physical layer is split in two parts. The functionality of sending and receiving packets on the air-channel is simulated by using a function model on packet level granularity. The actual steps of processing the packet such as preamble decoding and payload decoding are modeled on task graph granularity. Both the task graphs and their mapping to a target platform are part of the task graph level timing model which simulates timing at a finer granularity and signals the completion of processing steps back to the function model. A scheduler resides between the function model and the timing model and translates packet processing steps into task graphs and platform mappings. It may choose from a pre-computed set of task graphs and mappings and can set data dependent timings for certain tasks. In this way, the effect of run-time mechanisms can be studied. Models of run-time mechanisms have control over the choice of alternative implementations and mappings, which they exercise according to input data and the hardware platform state.

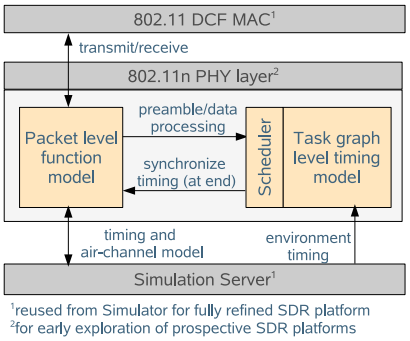

Fig. 1. Task graph timing model embedded in function simulation

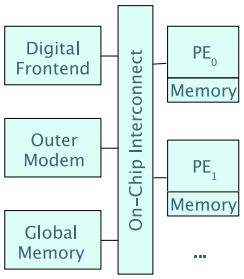

Fig. 2. Platform template

On top resides the 802.11 DCF MAC layer which controls timing for channel sharing and thus provides the physical layer with realistic input data and timing. The input for the MAC layer is either generated by a test-bench or by higher layers of the protocol stack.

Both the Simulation Server and the 802.11 DCF MAC layer are reused from another simulation framework developed at IMEC which intends to model a fully refined SDR platform [13]. The simulation framework presented in this paper, however, intends to explore prospective SDR platforms. Therefore, it has a completely different model of the 802.11n physical layer in order to enable both the exploration of various target platforms and the study of run-time mechanisms in an early development stage.

IV. Case Study – Platform Dimensioning

A. Experiments

a) Part 1 – Scheduler Evaluation: The first part of the case study is about analyzing a scheduling technique in a two protocol hand-over scenario where an error-rate/computation trade-off is used to reduce the number of required Processing Elements (PEs) from four to three.

In this experiment we assume an SDR platform design where the most computation intensive protocol requires more than 150% of the processing capability of one PE for highest quality decoding. As a consequence, the capability for a seamless hand-over – where two of these protocols are active for a short time – would require four PEs on the platform. However, the probability that both protocols require worst case computation efforts at the same time is not very high. Furthermore an error-rate/computation trade-off will be feasible during the short times when high-load phases of both protocols coincide. Due to this trade-off we assume that run-time management techniques can reduce the requirements of such a protocol to only one PE with a tolerable degradation in QoS. This has the potential to reduce the required number of PEs on the platform from four to three which can have a considerable cost impact.

The experiment of this part is about analyzing a simple run-time management technique used in the hand-over setup described above. What makes it extremely simple is that it greedily allocates one or two PEs per packet decoding task and does not support migration. Therefore we analyze the ratio of times when there is one PE or two PEs allocated for a protocol and the potential improvement in this ratio if the scheduler would support on-the-fly migration from one PE processing to two PE processing in the middle of a packet decoding task.

b) Part 2 – Interconnect Dimensioning: The second part of the case study is an analysis on finer granularity about on-chip interconnect traffic for a high load phase of decoding an OFDM packet.

The experiment of this part is to simulate payload processing of an OFDM protocol such as 802.11n with imaginary protocol rate. The payload processing is simulated on a platform with multiple PEs which communicate through an on-chip interconnect. The protocol rate is chosen as the rate that loads the PEs to 100% in order to dimension the interconnect such that it will not be the bottleneck in usual payload processing operation.

B. Platform models

The investigated platforms are based on the platform template shown in Figure 2. They consist of a set of PEs with local memories, a global memory, and two black box modules namely digital frontend and outer modem which are not the focus of this paper. The aforementioned components are connected by an on-chip interconnect like a bus, crossbar or Network-on-Chip (NoC). The PEs are powerful VLIW/Coarse Grain Array processors and are assumed to have only one control flow each. The validity of this simplification can be extended by characterizing the effects of executing certain task pairs simultaneously and applying techniques such as Software Multithreading [14]. The digital frontend is able to send and receive baseband samples on multiple antennas. Channel coding and decoding as well as CRC calculations are performed by the outer modem. The focus of this case study is on the baseband processing steps interfacing with samples of the digital frontend and channel coded data of the outer modem. The baseband processing is modeled as embedded software which is mapped on the available PEs.

C. Experimental Setup

We have been modeling the baseband processing software using a protocol based on 802.11n 40MHz drafts. We assume OFDM symbols to have 128 sub-carriers out of which 108 are used to transfer data. For encoding an OFDM symbol we use 160 samples of 16 bit in-phase and 16 bit quadrature-phase components each. In both parts we use QAM64 symbols with coding rate 3/4.

In the first part about scheduler evaluation we assume a two antenna MIMO reception and the normal protocol rate of 802.11n 40MHz drafts. We send 300 MTU sized packets (2312 bytes payload) through two channels to the receiver which decodes both streams on the same platform.

In the second part about interconnect dimensioning we use the four combinations of two or four antenna MIMO receptions being processed by two or four PEs. Payload decoding is scheduled on the PEs using data parallelization on

TABLE I
TRAFFIC CHARACTERIZATION

PEs	Ant.	Protocol rate	DFE → PEs	PEs → OMD
2	2	970 MBit/s	1.3 GB/s	970 MB/s
2	4	810 MBit/s	1.1 GB/s	810 MB/s
4	2	1.9 GBit/s	2.6 GB/s	1.9 GB/s
4	4	1.6 GBit/s	2.1 GB/s	1.6 GB/s

OFDM-Symbol granularity. Only the tracking part of payload decoding introduces an inter-OFDM-Symbol-dependency. The decoded data stream is forwarded to the outer modem. For the performance estimation of the task graph model, we assume PEs to be powerful VLIW/Coarse Grain Array processors of the ADRES [15] family. A recently taped out SDR platform hosts ADRES cores in 90nm technology with 16 function units per core running at 400MHz. For a prospective SDR platform in 45nm technology we assume a 2.5 fold performance increase per core.

D. Simulation Results

The results of the first part are measured with 300 MTU size packets being transmitted through two channels with the decoding tasks being scheduled on one platform with three PEs. The fraction of high quality (2 PE) decoded OFDM-Symbols is 65% for the first channel and 78% for the second channel. These fractions could be improved up to 84% and 90% if the scheduler would support migration. It is obvious that the scheduling algorithm is not fair. Percentages do not add-up to 100% due to back-off times on both channels. The simulation time on a Pentium M 1.8 GHz for simulating the decoding of 600 MTU size packets is 40 seconds. This is neither achievable with a full function model on task granularity nor using cycle accurate instruction set simulators.

The results of the second part are on a finer granularity. Due to the use of data parallelization, all samples provided by the digital frontend within a certain period are sent to one PE only. If we do not want to stress the latency of our PHY layer implementation, it is not necessary for the interconnect to provide independent links from the digital frontend to different PEs. Thus, the resulting traffic pattern for communication from the digital frontend to PEs is data bursts in round robin order. The size of each burst is 160 samples $*$ 4 bytes per sample $*$ the number of antennas.

The traffic pattern for the communication from the PEs to the outer modem is also characterized by round robin bursts. For each OFDM symbol the baseband processing on the PEs produces $108*6$ soft-bits. Each soft-bit is encoded with 6 bits. The burst size is thus 486 bytes $*$ number of antennas.

The simulation runs show that the inter-symbol dependency of the tracking task does not prevent considerable speedup of the data parallelization for up to 4 PEs. Table I summarizes the traffic characterization. There are four test cases resulting from the use of two or four antennas and two or four PEs. In each case the physical layer data rate of the protocol has been chosen just high enough to load the PEs to 100%. This rate is listed as "Protocol rate" and ranges from 810 MBit/s to 1.9 GBit/s. For each test case the total bandwidth of the round robin bursts from the digital frontend to the PEs (DFE → PEs) is listed as well as the total bandwidth of the round robin bursts from the PEs to the outer modem (PEs → OMD).

V. CONCLUSION

In this paper, we propose a modeling framework that enables the simulation of complex, dynamic hardware/software SDR designs. In order to enable both the exploration of various target platforms and the study of run-time mechanisms in an early development stage, a function model and a timing model have been developed and coupled with an 802.11 MAC layer to retrieve realistic input stimuli. The presented case study shows that the framework can be used to derive simulation results both on packet level granularity and on finer granularity by using a task graph based timing model. This can be used to investigate the impact of run-time management techniques in an early stage in the development process.

ACKNOWLEDGMENT

This work is partially supported by the E.C. funded program MOSART IST-215244,
`http://www.mosart-project.org`

REFERENCES

[1] F. Catthoor et al., "Custom Memory Management Methodology: Exploration of Memory Organization for Embedded Multimedia System Design," Kluwer Academic Pub., 1998.
[2] B. Bougard, S. Pollin, A. Dejonghe, F. Catthoor, and W. Dehaene, "Cross-layer power management in wireless networks and consequences on system-level architecture," in Signal Processing, Vol. 86, No. 8, p. 1792-1803, 2006.
[3] G. Goessler and A. L. Sangiovanni-Vincentelli, "Compositional Modeling in Metropolis," in Proc. EMSOFT'02, p. 93-107, 2002.
[4] E. A. Lee and Y. Xiong, "A Behavioral Type System and Its Application in Ptolemy II," in Formal Aspects of Computing, Vol. 16, No. 3, 2004.
[5] T. Schattkowsky and W. Muller, "Model-Based Design of Embedded Systems," in Proc. ISORC'04, p. 121-128, 2004.
[6] Object Management Group, "UML Profile for Modeling and Analysis of Real-Time and Embedded systems (MARTE)," Request for proposals. OMG document: realtime/05-02-06. http://www.omg.org/cgi-bin/doc?realtime/05-02-06.
[7] G. De Micheli and L. Benini, "Networks on Chip: A New Paradigm for Systems on Chip Design," in Proc. DATE'02, p. 418-419, 2002.
[8] J. Hu, "Design methodologies for application specific networks-on-chip," Ph.D. dissertation, Carnegie Mellon Univ., 2005.
[9] A. Jantsch, "Models of computation for networks on chip," in Proc. ACSD'06, p. 165-178, 2006.
[10] L. Benini et al., "SystemC Cosimulation and Emulation of Multiprocessor SoC Designs," in IEEE Computer, Volume: 36 Issue: 4, April 2003, p. 53-59, 2003.
[11] CoWare Inc., "Platform Architect," http://www.coware.com/products/platformarchitect.php, April 2008.
[12] S. Mahadevan, M. Storgaard, J. Madsen, and K. Virk, "ARTS: A System-Level Framework for Modeling MPSoC Components and Analysis of their Causality," in Proc. MASCOTS'05, p. 480-483, 2005.
[13] J. Declerck et al., "A Software Development and Validation Framework for SDR Platforms," in Proc. SDRForum'08, 2008.
[14] D. Scarpazza, P. Raghavan, D. Novo, F. Catthoor, and D. Verkest, "Software Simultaneous Multi-Threading, a Technique to Exploit Task-Level Parallelism to Improve Instruction- and Data-Level Parallelism," in Proc. PATMOS'06, p. 12-23, 2006
[15] B. Mei, S. Vernalde, D. Verkest, H. De Man, and R. Lauwereins, "ADRES: An Architecture with Tightly Coupled VLIW Processor and Coarse-Grained Reconfigurable Matrix," in Field-Programmable Logic and Applications, Springer Berlin/Heidelberg, p. 61-70, 2003.