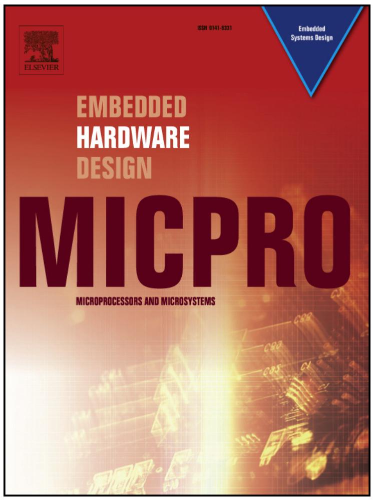


The COMPLEX reference framework for HW/SW Co-Design and Power Management Supporting Platform-Based Design-Space Exploration

Kim Grüttner, Philipp A. Hartmann, Kai Hylla, Sven Rosinger, Wolfgang Nebel, Fernando Herrera, Eugenio Villar, Carlo Brandolese, William Fornaciari, Gianluca Palermo, Chantal Ykman-Couvreur, Davide Quaglia, Francisco Ferrero, Raúl Valencia

PII: S0141-9331(13)00122-1
DOI: http://dx.doi.org/10.1016/j.micpro.2013.09.001
Reference: MICPRO 2072

To appear in: *Microprocessors and Microsystems*

Please cite this article as: K. Grüttner, P.A. Hartmann, K. Hylla, S. Rosinger, W. Nebel, F. Herrera, E. Villar, C. Brandolese, W. Fornaciari, G. Palermo, C. Ykman-Couvreur, D. Quaglia, F. Ferrero, R. Valencia, The COMPLEX reference framework for HW/SW Co-Design and Power Management Supporting Platform-Based Design-Space Exploration, *Microprocessors and Microsystems* (2013), doi: http://dx.doi.org/10.1016/j.micpro.2013.09.001

The COMPLEX reference framework for HW/SW Co-Design and Power Management Supporting Platform-Based Design-Space Exploration

Kim Grüttner[a], Philipp A. Hartmann[a], Kai Hylla[a], Sven Rosinger[a], Wolfgang Nebel[b], Fernando Herrera[c], Eugenio Villar[c], Carlo Brandolese[d], William Fornaciari[d], Gianluca Palermo[d], Chantal Ykman-Couvreur[e], Davide Quaglia[f], Francisco Ferrero[g], Raúl Valencia[g]

[a]*OFFIS – Institute for Information Technology, Oldenburg, Germany*
[b]*Carl von Ossietzky University Oldenburg, Germany*
[c]*Universidad de Cantabria, Santander, Cantabria, Spain*
[d]*Politecnico di Milano, Italy*
[e]*IMEC, Belgium*
[f]*EDALab s.r.l. - Networked Embedded Systems, Verona, Italy*
[g]*GMV Aerospace and Defence S.A.U - Madrid, Spain*

Abstract

The consideration of an embedded device's power consumption and its management is increasingly important nowadays. Currently, it is not easily possible to integrate power information already during the platform exploration phase. In this paper, we discuss the design challenges of today's heterogeneous HW/SW systems regarding power *and* complexity, both for platform vendors as well as system integrators.

As a result, we propose a reference framework and design flow concept that combines system-level power optimization techniques with platform-based rapid prototyping. Virtual executable prototypes are generated from MARTE/UML and functional C/C++ descriptions, which then allows to study different platforms, mapping alternatives, and power management strategies.

Our proposed flow combines system-level timing and power estimation techniques available in commercial tools with platform-based rapid prototyping. We propose an efficient code annotation technique for timing and power properties enabling fast host execution as well as adaptive collection of power traces. Combined with a flexible design-space exploration (DSE) approach our flow allows a trade-off analysis between different platforms, mapping alternatives, and optimization techniques, based on domain-specific workload scenarios. The proposed framework and design flow has been implemented in the COMPLEX FP7 European integrated project.

Keywords:
HW/SW Timing and Power Estimation, Virtual Prototyping, Design-Space Exploration, Power Management

1. Introduction

Increasing computing power and shrinking sizes of processing elements allow more and more functionality to be realized within embedded systems. In earlier times embedded systems have only implemented simple functions, but today complete systems, also known as System-on-a-Chip (SoC), can be implemented on a single chip. In the past a lot of effort has been spent on characterizing and estimating hard- as well as software parts of the SoC [1, 2]. For both of them relatively good techniques and tools exist, hence their properties and characteristics are well known and well understood.

In large and complex SoCs, components of the system, such as hardware, software, memory, and interconnect, cannot be considered separately. They must be considered together while interacting to capture the entire system behavior [3, 4, 5]. For analysis of the dynamic behavior this implies a behavioral simulation of all system components and their interaction, also with their environment. This is essential for power and tim-

Email addresses: `gruettner@offis.de` (Kim Grüttner), `nebel@informatik.uni-oldenburg.de` (Wolfgang Nebel), `evillar@teisa.unican.es` (Eugenio Villar), `fornacia@elet.polimi.it` (William Fornaciari), `ykman@imec.be` (Chantal Ykman-Couvreur), `davide.quaglia@edalab.it` (Davide Quaglia), `fferrero@gmv.com` (Francisco Ferrero)

ing estimations. For specific platforms, proprietary simulation environments are available for both timing and power models (see Section 2). But a common and open framework, suitable for a large range of platforms and designs is still missing. Such a framework would allow comparing different platform characteristics and thus rapid prototyping and design space exploration. Performance bottlenecks and power peaks within the entire system could be identified in early design phases, where modifications of the system are easier and more affordable than in later phases.

In this paper we present a new design flow concept and propose a framework that combines system-level power optimization techniques with platform-based rapid prototyping. For the proposed approach we focus on the derivation of an executable virtual prototype of an embedded HW/SW systems right after functional specification, partitioning, and mapping to an implementation platform. Execution or simulations of this prototype allow decisions concerning architecture, performance, power, and price, early in the development process. The challenges we need to face in today's Electronic System Level (ESL) design-flow, including existing point tools that are able to address single design issues, are presented in Section 2. The proposed framework, that is the result of the COMPLEX European project [6], follows a unified system-level specification for HW and SW, but utilizes different estimation techniques for custom HW and SW, as well as pre-defined IP components. Since good stand-alone solutions for each of these parts exist, Section 3 presents our concept that unifies state-of-the-art tools and techniques in a common design-flow. Section 4 presents different industrial use-case aware customizations of the proposed design-flow. Conclusions are drawn in Section 5, which sums up our concept and points out following work.

2. Requirements, existing tools & related work

This section aims at defining requirements (highlighted as **R1-8**) for early system specification, modeling, and simulation at ESL. In this sense, we are giving a brief overview of commercial tools and related work.

Generally, embedded system design is a sequence of design decisions that finally leads to an implementation. These decisions are made with regard to the functional and extra-functional system requirements. These need to be captured and tracked during the entire design flow. All dimensions (e. g. functional, timing, power, performance, memory, area, etc.) of the design space need to be explored in order to estimate the costs of the final system implementation. For this purpose a dependable and accurate model of the future system and workload scenario, prior to heavy investments in HW and SW development is necessary. In our approach we focus on the enrichment of virtual execution platforms with extra-functional properties. In the proposed approach, we do not address an automated path to implementation or automatic synthesis of physical platforms, yet.

Today's virtual prototyping platforms are provided for early software development and functional testing. These platforms are often built using SystemC TLM-2.0 interfaces at AT (Approximately Timed) or LT (Loosely Timed) abstraction. Today these platforms are used for functional software development and software stack configuration. Most virtual platforms are targeting mainly functional and loose timing aspects in early software development. Power dissipation or the functional effects of dynamic power management strategies cannot be taken into consideration, and thus cannot be explored at early design stages.

Code generation in Model-Driven Engineering (MDE), high-level synthesis techniques for custom HW, and abstraction of existing components to system-level descriptions are key enablers for unified system-level simulation approaches [7] and need to be urgently combined with the virtual prototyping platform approach as mentioned above. We want to *consider functional, power and timing behavior on system-level under explicit partitioning and mapping to a specific implementation platform* (**R1**). This requires:

1. a formal system-level specification to be linked with functional and extra-functional requirements to be checked during the entire design process,
2. an executable model that can be derived automatically from the formal specification model, which is capable to simulate large sequences of domain-specific workloads in an acceptable time,
3. the representation and annotation of extra-functional information (timing and power) in the executable specification.

For today's embedded HW/SW systems this cannot be done on RT-level anymore, due to high simulation complexity. The requirements of the proposed framework [8], along with point-tools (as described in [9]) that fulfill some of these requirements, are subdivided into four categories discussed below:

2.1. MDE and Executable Specification

The complexity and size of current systems are increasing, which may well lead to longer development times and integration challenges. In the contrary, time-to-market constraints are tighter and there is a larger incidence of extra-functional demands (i.e. power consumption, real-time behavior). Therefore, it is necessary to provide mechanisms that permit the separation of the functional and extra-functional concerns in order to better master the complexity of current embedded systems. The adoption of the Model-Driven Engineering (MDE) principles improves the separation of concerns in the specification, i.e. functional from extra-functional concerns, and the application from the platform [10]. It enables the distribution of the specification task among designers, and a parallel development of the specification, e.g. one designer can describe the application, while other, the platform. *The designer should operate on an executable specification that enables the same functional description for HW and SW* (**R2**). This is permitted by definition of a common specification language at system level, a unique, well-defined parallel programming paradigm and the formulation of HW/SW partitioning through different platform mappings. *The communication model should be abstracted from platform communication resources and enable the remapping to different communication infrastructures* (**R3**).

Following a component-based approach is highly desirable in a HW/SW design as it improves the product organization, its reusability and modularity. In this approach, the component represents an elemental unit for architectural mappings, which simplifies the analysis of the system extra-functional attributes and facilitates the design space exploration.

The system specification follows the platform-based design (PBD) approach [11, 12], that separates the application, the hardware platform and the mapping of the application to resources of the hardware platform. This is compatible to the MDE principles, known from the area of software engineering, where the specification of the system's functionality and the platform should be done separately. The functional properties of the system should be specified in the application specification. Extra-functional properties like the deadline of application tasks or the use of different IP cores, target technology, maximum chip area, etc. which influence timing and power dissipation need to be specified separately. Some widely used commercial available tools and languages for parallel executable system specification are: LabView, Matlab/Simulink & Stateflow, Rhapsody & Statemate, Esterel SCADE, and C/C++/SystemC-based approaches. For adding extra-functional properties different UML profiles, like MARTE, have been proposed. All of these tools, modeling languages and profiles already offer a direct path to implementation for embedded HW/SW systems. However, the effects of these implementations in terms of timing behavior and power consumption can hardly be analyzed. [13] proposes an interesting approach to power efficient system design using UML generated executable models. The use of the MDE principles in the design of HW/SW systems and standard modeling languages like UML allows the definition of modeling methodologies that are based on the separation of functional and extra-functional concerns and the specification of both hardware and software system. This is convenient in a context where system engineers need to produce a specification of the embedded systems suitable for early design stages (e.g. functional verification, design space exploration). Indeed, it can also serve as an unambiguous design requirement specification for hardware and software designers in later phases of the design.

Moreover, the use of modeling languages like UML provides an unambiguous way of documenting the system architecture (i.e. both hardware and software) where a set of diagrams provide a mean to illustrate the different system facets that are for the interest of the designer. Through extensive tool support, from UML model editors to model-to-model and model-to-text, the automatic generation of system executable specifications is enabled. As it was mentioned above, such automation not only saves time, but also leverages a unified approach once the coherence between the UML model and the executable model can be better ensured.

However, UML itself does not provide the necessary semantics to model extra-functional properties. The use of the MARTE profile gives some of the required semantics (i.e. target technology, area, task deadlines), while others related to the DSE must be provided by means of UML profiling mechanisms.

Finally, modern dynamic performance estimation techniques (i.e., native simulation, virtualization) enable the generation of fast executable performance models, suitable for feeding a design exploration loop.

2.2. Estimation and Representation of Extra-Functional Properties

Following the PBD approach enables a separation of application and execution platform. The mapping describes a binding of the application to allocated execution, communication, and memory resources of the platform. All these mapping decisions have an influence on the system properties including the correct behavior in terms of timing, power, and costs.

In this paper we focus on timing and power dissipation aspects, referred to as implementation artifacts of the execution platform, consisting of processing elements, memories, and interconnect. As mentioned in the previous paragraph several tools exist that are able to generate a C/C++/SystemC implementation of the behavior of the system. This representation shall be a common starting point to perform analysis and code optimizations under consideration of constraints induced by the chosen target platform and user-defined parameters.

Current extra-functional property estimation techniques can be classified into four groups:

Simulation-after-synthesis-based methods: Hardware performance can be evaluated by simulating the synthesized description that a high-level synthesis tool generates [14]. During recent years, several commercial HLS tools have become available [15, 16, 17, 18]. These tools can synthesize a register transfer level (RTL) model from a behavioral description (usually a C/C++ code), although it is frequently necessary to manually rewrite the source code. After high-level synthesis with specific constraints/directives, the RTL description has to be simulated with a test-bench to obtain estimations. This approach (simulation-after-synthesis) produces very accurate results since a real hardware implementation is evaluated with a test-bench, thus it is normally used as a reference performance model by other hardware performance analysis techniques. However, it requires high execution times to synthesize and simulate the model with different constraints/directives.

Static performance analysis: Most hardware estimation works presented in the literature are based on static source code analysis [19, 20, 21]. In this type of approach, a Control and Data Flow Graph (CDFG) is normally generated from the behavioral system description. This graph models static data dependencies and enables the estimation of the execution time without simulation. However, software languages like C/C++ normally include constructions that may generate dynamic data dependencies (for example, pointers), thus only a grammar subset of these languages can be efficiently evaluated using static analysis techniques. Additionally, different synthesis constraints or input test patterns produce different estimation results, thus some static performance analysis methodologies provide estimation bounds instead of a single value [19, 21]. They typically generate an upper limit (Worst-Case Execution Time, WCET) and/or a lower limit (Best-Case Execution Time, BCET). Static performance analysis tools normally use simplified/relaxed versions of high-level synthesis algorithms or probabilistic approaches. They can also estimate area [21] and clock period [22]. Static performance estimation techniques typically provide speedups of about two orders of magnitude (x100) compared with the simulation-after-synthesis approach. The typical estimation error ranges from 20% to 30% [22].

Model-based approaches: Model-based estimation techniques use parameterized functions to evaluate hardware performance of pre-defined or Intellectual Property (IP) modules. They normally estimate area, power [23], clock cycle and state-number/latency [24]. These techniques provide very fast estimations because they only have to execute the estimation function with the particular configuration parameters of the current instantiation of the pre-defined/IP modules. Additionally, a very time consuming methodology that normally requires a very large number of synthesis processes is needed to derive the parameterized functions. Using mathematical techniques, such as linear regression and curve fitting techniques, these methodologies adjust the parameterized models with the estimation parameters. Some approaches [24] try to predict hardware performance of different types of hardware modules using a common methodology. The main disadvantage of these approaches is the high estimation error (between 69 % and 108 % [24]).

Trace-based techniques: Trace-based techniques can also be used to provide hardware performance analysis [25]. In this approach, a trace is generated during source-code execution (simulation-based approach). This trace is used to identify data-dependencies and operation sequences. After trace generation, an estimation tool schedules the operations using similar techniques to the static analysis approach but it can only estimate one possible implementation. This technique can support dynamic dependencies but it cannot be easily integrated into a co-simulation environment. Additionally, the trace size limits the use of this technique to low-medium size estimation problems.

The system's timing and power dissipation need to be studied under domain specific workload scenarios (**R4**). For the assessment of the system's overall power dissipation, average or typical cases are usually more important than worst case workloads. Concerning real-time requirements worst-case execution times (WCET) need to be studied instead of average case execution times. *For an overall system characterization both kinds of analysis need to be performed for the software, custom hardware, communication, and pre-existing IP components* (**R5**).

Power and timing estimation of full-custom HW (ASIC) (**R5.1**) depends on several design parameters like micro-architecture size and design, level of parallel processing, layout implications but it also depends on run-time characteristics like activity, frequency, or dynamic power management policies. Transistor technology also has major impact on dynamic and static power values as well as on timing. Additionally, the environmental temperature has major impact on HW's power and in extreme case may force the chip to slow down. The timing and power estimation of ASIC hardware under consideration of these aspects have gained in importance and are addressed by today's industrial high-level synthesis tools, such as [16, 17, 18, 15].

Pre-existing hardware IP component's (**R5.2**) extra-functional properties are hard to obtain. If available these properties can be taken from a data sheet or in the worst case they need to be estimated manually. Due to IP protection this can become a time consuming task. After estimation the next challenge is to represent and back-annotate these extra-functional properties in executable functional models, representing the corresponding IP blocks in an overall system simulation. For this purpose power state machines can be used, but nevertheless these models are neither standardized nor part of today's system-level design languages and ESL tools.

Estimation of embedded software's power and timing (**R5.3**) is complicated due to processor attributes like pipelining, branch prediction, superscalar architectures, and out of order execution. Additionally, power and timing of the processor depends on external influences such as memory stalls, interrupts, task switches, and operation system calls. To cover these aspects during power estimation vendor specific or generic instruction set simulators (ISS) as ARMulator or SimpleScalar with augmented power models are used. For SW WCET analysis static analytical methods are used instead of simulative approaches. AbsInt's aiT WCET analysis tool is used by the industry and allows easy integration with existing executable specification environments like Esterel's SCADE Suite.

Estimation of extra-functional properties for HW/SW communication (**R5.4**) depends on the implemented functionality, the partitioning into embedded SW and custom HW and its mapping to execution units, and finally the data dependencies induced by the provided use-case. Using the SystemC-TLM methodology allows vendor specific provision of bus models. These models can be implemented on different levels of granularity, from abstract atomic transitions, over the consideration of request and transfer phases, down to a bus-cycle accurate view. *Timing of communication* should be encapsulated in dedicated communication models and should reflect the communication protocol's behavior as well as data dependent effects, such as transaction lengths. For an appropriate *communication power model* the situation is more difficult, since the vendor of the bus model does not necessarily have information about its implementation and its influencing parameters for the power model. To this end the bus vendor will not be able to provide a power model, but the API of the bus-functional model shall offer extension points for the platform developer.

A realistic system contains all these types of components. They must be considered together to observe the influences of their interaction (**R6**). Thus, a holistic estimation combines individual estimation results for embedded SW, custom HW, IP components, and HW/SW communication, to allow a simulation and estimation of the overall system behavior and its extra-functional properties. Such an approach uses existing and highly specialized tools to characterize individual parts of the system. Typical estimation approaches, as mentioned above, provide models for the underlying HW. The challenge is to use information from these low-level models for creating higher-level models that can be combined with the behavior in a functional simulation.

The COMPLEX reference framework enables the combination of different system components and describe a set of alternative power models for each.

Embedded processors: Several techniques have been developed for modeling processor power consumption at the system level. The complexity of these models vary, depending on the volume of, and frequency with which, information is extracted from a simulation model of the processor. Table 1 lists several alternatives, in decreasing order of computational effort. In the first model, for every clock cycle, the complete pipeline state of the processor is captured, and the combination of instructions found in the different stages is used to estimate power consumption [26, 27]. In the second model, for each cycle, only the instruction that is currently being executed is extracted [28]. In the third model, over discrete time intervals, only the number of instructions of different predefined types is counted to compute total energy or average power [28]. The fourth approach, software energy macro-modeling, involves monitoring code sequences of larger granularity (e.g., function calls) [29]. The fifth and simplest power model we consider is based on parameters such as power modes, operating voltage, and frequency [28].

On-Chip Buses: Numerous models have been proposed for estimating the power consumption of global buses. Examples of such power models that we have considered in our framework are listed in Table 1. In the first approach, on every cycle, transition activity is examined on individual bus lines, and is used to estimate power, using transmission line models that capture deep sub-micron effects, and effects of the drivers and repeaters [30]. In the second model, for each cycle, aggregate transition activity is used to estimate power consumed on global buses, using a lumped capacitance to model driver, repeater, line, and parasitic capacitances. The third model is an analytical one, in which over a certain time interval, the number and types of bus transactions are monitored, and used to estimate average transition activity, which can then be used to estimate average power.

Caches/Memories: Cache power models include those that are targeted towards cycle-level simulation environments [26], as well as more efficient analytical models that are targeted towards exploring alternative cache architectures [31, 32]. For our framework, we consider the two models listed in Table 1. In the first model, on every access to the cache, the power consumed by the cache is computed based on the type of access (read/write), the result of the access (hit/miss), and transition activity on the bit and word lines. In the second model, over a certain time interval, statistics that capture the number and types of cache accesses are used as inputs to an analytical model that computes average cache power, using lumped capacitances for different cache components, and estimated transition activity.

Custom Hardware: Power analysis of hardware, including both custom hardware, as well as standard components such as memory controllers, timers, and other peripherals, has traditionally been performed at the logic- and register-transfer levels (RTL). Recently, advances have been made in estimating the power consumed at the cycle-accurate functional and behavioral levels [33, 34, 35]. While each abstraction level in itself represents a potential trade-off between power estimation accuracy and computational effort, approaches based on logic-/RT-level power estimation are unacceptably slow for system simulation.

2.3. System Simulation Including Extra-Functional Properties

Fast system simulation is important due to the raising system complexity (**R7**). Today most state-of-the-art mobile embedded systems like phones, cameras, music and video players are using complex HW and SW, including operating systems and middleware. Virtual prototypes that are used for the simulation of these systems can no longer be executed on a cycle-accurate ISS connected via a cycle accurate bus model to a register transfer (RT) level representation of custom HW. To boost simulation performance virtual platform simulators allow to abstract from certain HW platform details, like the cycle-accurate representation of a processor's

HW component		Power model	Accu.	Eff.
Processor	1	Pipeline state aware	++	- -
	2	Instruction-level	+	-
	3	Analytical, Instruction-class based	o	o
	4	Function-level macro-models	-	+
	5	Mode-based	- -	++
Interconnect	1	Distributed RC models	+	-
	2	Lumped capacitance	o	o
	3	Analytical, transaction based	-	+
Cache/memory	1	Structure-aware, access level	+	-
	2	Analytical, access-statistics based	-	+
Hardware	1	Gate-level	+	-
	2	Register-transfer level	o	o
	3	Cycle-accurate functional level	-	+

Table 1: Power models for system-level power estimation (Accu. = Accuracy, Eff. = Effectiveness in terms of (simulation) speed)

instruction set or cycle accurate communication models. Some works have been proposed in the past for translating RTL VHDL and Verilog models into C/C++ descriptions, targeting verification of HW models via simulation [36, 37, 38, 39, 40, 41, 42]. Furthermore, today's virtual platform simulators make use of native host system execution to speed-up simulation [43]. Concerning the representation of extra-functional properties, related work is mainly focused on power analysis. In [44], Power State Machines have been proposed for the first time in system-level modelling. Since dynamic energy consumption emerges from activity, in [45] a concept is presented where activity is observed at the communication interfaces. With *Powersim* [46] the model does not have to be changed because a modified SystemC simulation kernel is used. In [47] a system-level model that separates architecture and functional application model is presented. The application model is mapped on the platform model, which includes a model for the energy consumption. Nevertheless, the raise of abstraction and its gain in execution speed comes at the cost of a loss in accuracy. But depending on the application characteristics, workload scenarios, and of course the system issue under investigation, a certain loss in accuracy is acceptable. A simulation framework should offer the possibility to switch between different simulation granularities for both power and timing. Commercial available simulation platforms are ARM RealView & MaxSim, Mentor Graphics Vista, Synopsys Virtualizer & CoMET and METeor, Cadence System Development Suite, Wind River Simics and Open Virtual Platform [48]. Concerning networked embedded systems no appropriate tools are available in the ESL domain to simulate also the networking aspects. Currently designers are forced to use co-simulation with pure network simulators like NS-2 [49], OPNET [50] or OMNET [51]; the co-simulation slows down the entire simulation.

2.4. Design-Space Exploration

Platform-based design approach is widely used to meet time-to-market constraints [11]. In this scenario, the basis for designing novel systems is to have platform with lots of parameters that can be exposed to the designer to enable finding the best HW/SW architecture that meets the application requirements. The tuning phase of platform configuration, hardware-/software partitioning, scheduling, and mapping on the platform resources is called *Design Space Exploration* (DSE).

In the past, this phase has been mainly done by using manual approaches guided by the designer experience. Nowadays this possibility is becoming unfeasible for two main reasons: the first deals with the time needed for the exploration phase, while the second one with the performance unpredictability of novel parallel architectures.

Regarding the first reason, the exploration phase should be guided, but cannot be performed by the designer because it requires too much time. In computer architecture research and development, simulation still represents the main tool to predict the performance of alternative architectural design points. If we consider a cycle-accurate system-level simulation, it requires a lot of simulation time and this time will continue to grow due to the increment of system complexities [52]. For this reason, the exploration of the design alternatives can exceed the practical limits even if the number is not so large [53]. *This means that DSE requires being automatic and efficient in terms of optimization algorithms and predictive models, reducing the number of simulations to be done and thus saving time* (**R8**). Moreover, novel HW/SW architectures continue to grow both in terms of number of interacting components and in terms of tunable parameters exposed to the designer. This makes the impact of novel parameter configurations difficult to predict, even when the designer knows the target applications and architecture. Furthermore, multiple competing objectives (e.g. Power and Performance) result in searching for multi-objective optima (Pareto solutions). In this context, the key challenges are to support the designer in a more systematic way through the design space exploration phase, in particular: 1st by decoupling the simulation infrastructure from the design space exploration environment with analysis support, 2nd to efficiently face the DSE search by using novel optimization strategies coupled with predictive models, and 3rd to combine exploration of hardware and software parameters considering both the design-time and run-time tuning of the target platform. Despite of some effort have been spent from the community in all the three directions, such as [54, 55], [53, 56, 57] and [58, 59] respectively, the problem is far to be solved.

2.5. Integrated Framework

To the best of our knowledge, there is no integrated framework available that enables the estimation and simulative analysis of timing and power properties from a HW/SW independent executable specification model under manual mapping constraints, as described above. The goals of our presented approach are:

- Interface to model-driven SW design entry using MARTE/UML and the industry standard Matlab/Stateflow model-based design environment,
- combination and augmentation of well-established commercial ESL synthesis and analysis tools into a seamless design flow enabling performance and power aware virtual prototyping from a combined HW/SW perspective,
- fast simulation and assessment of functional and extra-functional (time and power) properties of the entire system after platform mapping with scalable accuracy,
- framework for dynamic adaptation to changing context and transparent optimization of platform resource usage,
- multi-objective HW/SW co-exploration to assess the design quality and to optimize the system platform with respect to extra-functional properties.

3. Proposed concept

The design framework proposed in the COMPLEX project is illustrated in Figure 1. As presented in Section 2 we follow the PBD approach with a separation of application ⓐ/ⓔ, architecture ⓓ/ⓖ, and mapping description ⓒ. The architecture/platform consists of pre-existing IP components like processors, buses, hardware accelerators and memories, while the application describes how these resources are used to implement certain system functionality. For the specification of different domain-specific application workload scenarios,

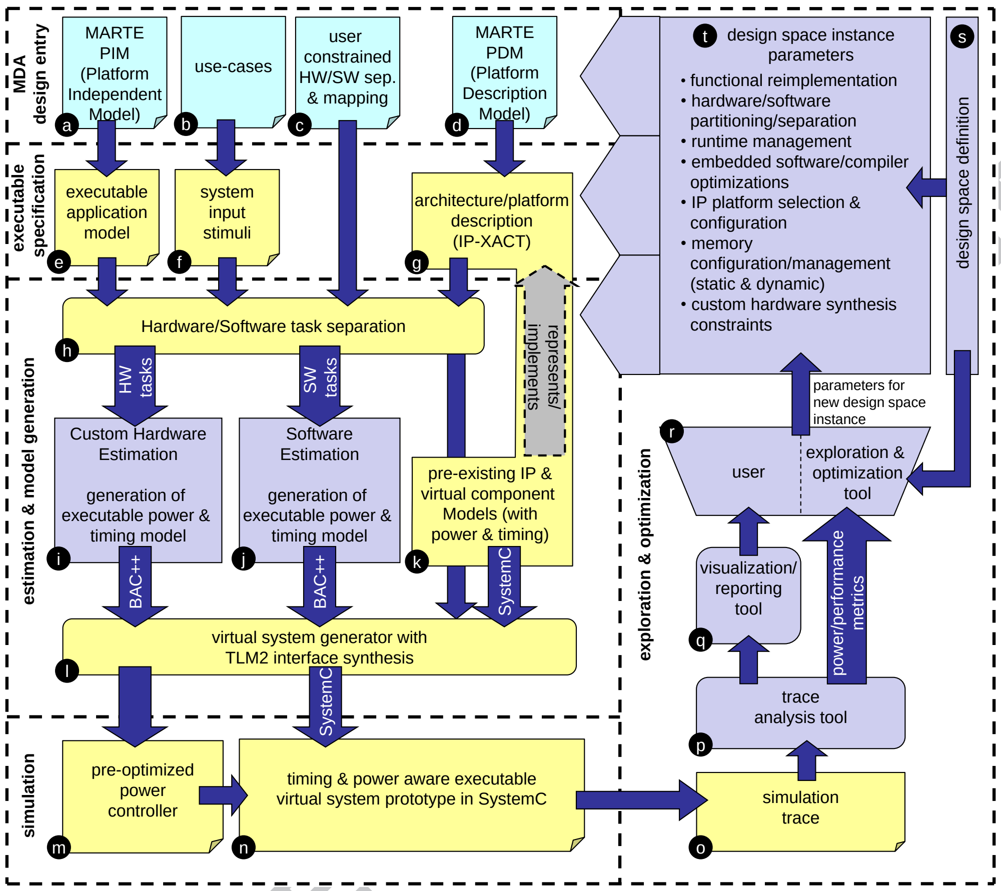

MDA design entry
MARTE PIM (Platform Independent Model)
use-cases
user constrained HW/SW sep. & mapping
MARTE PDM (Platform Description Model)
executable specification
executable application model
system input stimuli
architecture/platform description (IP-XACT)
estimation & model generation
Hardware/Software task separation
HW tasks
SW tasks
represents/ implements
Custom Hardware Estimation
generation of executable power & timing model
Software Estimation
generation of executable power & timing model
pre-existing IP & virtual component Models (with power & timing)
BAC++
BAC++
SystemC
virtual system generator with TLM2 interface synthesis
SystemC
simulation
pre-optimized power controller
timing & power aware executable virtual system prototype in SystemC
simulation trace
trace analysis tool
visualization/ reporting tool
power/performance metrics
user
exploration & optimization tool
exploration & optimization
parameters for new design space instance
design space instance parameters
• functional reimplementation
• hardware/software partitioning/separation
• runtime management
• embedded software/compiler optimizations
• IP platform selection & configuration
• memory configuration/management (static & dynamic)
• custom hardware synthesis constraints
design space definition
a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p
q
r
s
t

Figure 1: The COMPLEX Reference Framework

specified as use-cases ⓑ we propose to generate a system input stimuli specification ⓕ for triggering the executable system model.

The most important property of the proposed framework is that *timing and power characterization is separated from application specification and development*. This separation allows platform providers to offer timing and power characterized virtual platform component models (IPs) ⓚ. Together with the estimated custom HW ⓘ and SW components ⓙ a timing and power aware executable virtual system prototype ⓝ can be generated.

Based on the simulation trace ⓞ, obtained from executing the generated platform model, analysis tools ⓟ can either generate a report or a visualization of the power consumption per system component over time ⓠ. The application of metrics on the trace is used to drive an automatic or semi-automatic exploration and optimization process ⓡ that modifies different design parameters ⓣ in a pre-defined design-space ⓢ. These parameters can be applied on the MDA design entry model, executable SystemC model, or the estimation and model generation tools.

The following sections give a more detailed description the different phases of our proposed rapid prototyping framework.

3.1. MDA Design Entry

The COMPLEX modeling entry is supported by the COMPLEX UML/MARTE modeling methodology [60, 61] that includes a tool set fully integrated in the Eclipse framework [62]. This tool set automates the generation of the code which serves to generate the performance executable model. The UML/MARTE specification models both the system and the input stimuli environment. Among all the features, the following will be described in the next paragraphs:

- *Viewpoints* for separation of functional and extra-functional concerns
- *Component-based design* approach

- Explicit support for *Design-Space Exploration* (DSE)

Viewpoints. The COMPLEX UML/MARTE methodology enables the specification of the different facets of the system in different model *viewpoints*. The COMPLEX model viewpoints are the *Data Model View*, the *Functional View*, the *Communications and Concurrency (CC) View*, the *Platform View*, the *Architectural View*, and the *Verification View*. These views enable separation of concerns and thus raise the level of abstraction as each view focuses on a specific aspect of interest of the system. The COMPLEX UML/MARTE methodology also defines the relationships among these views, and a workflow which guarantees them. This enables the designer to build a synthetic model (avoiding possible redundancies and thus coherence checks) and enables a cooperative workflow where the application and platform can be captured in parallel. The separation of concerns is given at several levels of the model design. The system (i.e. application mapped onto platform resources) is separated from the environment. Within the system model, the platform specification (i.e. processing resources, operating system) is separated from the model of the application. Finally, within the application, data structures, functionality (interfaces and classes) and application components are also separately captured. The extra-functional properties of the application are specified on the *CC view*, while the platform extra-functional properties are described in the *Platform view*. The *Architectural view* provides information about the allocation of application components onto the platform components and the DSE parameters and metrics can be reflected.

Component-based design. The COMPLEX UML/-MARTE methodology follows also a component-based approach. At the application level, the designer packets the functionality within application components in the *CC View*. A system application is captured as a component, and instances of application components are used to capture the application architecture. This component represents the Platform Independent Model (PIM) according to the MDA paradigm. The platform architecture is captured in the *Platform view* by means of instances of SW and HW platform components, e.g. RTOS component instances, and instances of HW processor components. This architecture represents the Platform Description Model (PDM) according to the MDA paradigm. The component is the elemental unit used in the COMPLEX UML/MARTE methodology for deploying functions onto processing resources (i.e. microprocessors, FPGA) in the *Architectural view*. For instance, application components can be mapped onto the platform components which represent processing resources.

Use-cases, scenarios, verification. Finally, the COMPLEX UML/MARTE methodology allows designers to specify the input stimuli in a separated view, the *Verification view*. As mentioned, this view supports the specification of a set of environmental components and how they connect with the system component. This view provides a description of the interactions (through sequence diagrams) among the environmental components and the system component as the sequence of ordered messages in the context of a use case scenario. Timing information and ordering constraints among environment events are captured in the sequence diagrams, so that it enables the documentation and generation of realistic use cases. This is crucial for the dynamic performance estimation enabled by the executable model derived from the UML/MARTE system model. The methodology enables the definition of multiple scenarios that represent different use cases of the system. Designers may choose any scenario from those modeled in the *Verification view* to generate the performance executable model and explore the design space.

Support of Design-Space Exploration. The COMPLEX UML/MARTE methodology has been explicitly designed for supporting design space exploration. Specifically, the methodology supports the specification of a design space, i.e., a set of design solutions, rather than a single design. The description of such design space is enabled by means of defining: a set of architectural mappings (*allocation space*),a range of values for platform attributes (*parameters of the space*), and several platform architectures (*architecture space*).

In order to specify this design space, the methodology relies on the MARTE profile and proposes new stereotypes for the missing semantics (i.e. DSE, IP-XACT concepts), which are proposed as a necessary enhancement of current capabilities of MARTE for embedded system design. Moreover, the COMPLEX UML/-MARTE methodology supports also explicit constraints and rules that can be used by the designers to limit the space of solutions to those that are of main interest. The methodology also supports the specification of system local metrics. In contrast to global system metrics, such as total power consumption, system dependent metrics depend on each specific system model. For example, the latency metric for servicing a specific function of an application component or the miss rate of the instruction cache of a given processor of the platform. This feature provides a capability of paramount importance: all the aspects of the system with some impact on its final performance: application and platform, SW and HW, architectures and architectural mappings, component attributes and different types of metrics, form now part of the DSE loop and therefore can be optimized at once, under a real holistic approach.

3.2. Executable Specification

The output of the MDA entry is an executable model generated from the PIM. The PDM is used to generate a structural platform model with virtual processing, memory, and communication elements. These are used to model the resource constraints of the execution platform. The generated executable specification (Algorithm Domain), platform description model (Architecture Domain), and platform mapping are shown in Figure 2(a).

Executable application model. In our executable application description model we perform a separation of behavior (computation) and protocol (communication). Our concurrent building blocks are tasks or processes that contain a behavioral and a protocol part. The *behavior* part describes the function or algorithm to be executed, written in sequential C/C++ code. This description is independent from an implementation in either HW or SW. Behaviors can be composed of functions and describe a pure sequential execution order. This enables reuse of existing software descriptions. Moreover, the tools mentioned in Section 2.1 allow synthesis to a C/C++ representation. An *abstract task* describes a "'Runnable"' i. e. a process. Each abstract task contains a single behavior. Abstract tasks can either be active or passive. An *active task* starts running immediately after its activation and can either be blocked through a communication request or when its computation is finished. An active task can be (self) triggered again after a certain amount of time (time triggered task, or periodic task). *Passive tasks* can only be triggered by active tasks through explicit requests. A passive task cannot trigger itself and it cannot trigger any other passive task.

The *protocol* part describes communication among behaviors. It is realized through a port that allows active tasks to call service on another passive task's behavior. These calls are blocking, i. e. the caller's behavior can be continued after the service call has been completed. When multiple active tasks are requesting a service call of the same passive task a scheduling action is required. More details about this can be found in [63, 64]. These service calls abstract from a certain communication protocol implementation in either HW or SW.

System input stimuli. In order to examine and analyze the parallel application description model under a certain workload scenario, it needs to be stimulated accordingly. The system stimuli might originate from user interaction or communication with other components that are part of the system's environment. These stimuli describe use-case scenarios and can be derived from a UML/MARTE use-case specification or from an environment model in Matlab/Simulink.

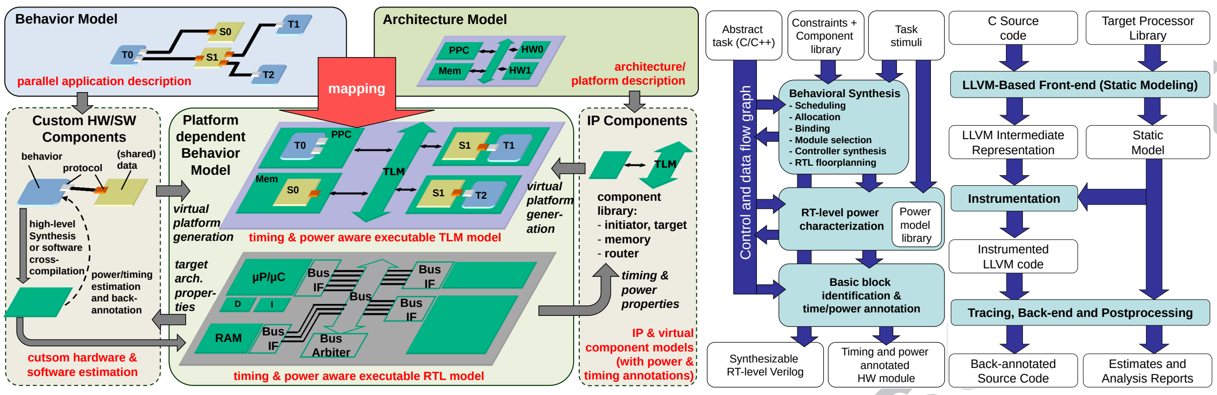
User constrained HW/SW separation & mapping. The user-constrained HW/SW separation and mapping defines the binding of tasks from the application model to execution resources of the architecture/platform description model. Active and passive tasks can be mapped to execution resources, while passive tasks can only be mapped to memories.

Architecture/Platform description. The platform description model is composed independently from the application model. It is a pure structural and extra-executable representation of the execution platform consisting of execution resources (like SW processors, DSPs or ASICs), memories, communication resources (like shared buses), and pre-existing IP components. In addition, constraints that have a direct influence on the timing and power consumption are represented in the component's meta-description. For SW processors it is the instruction set architecture (ISA) including its pipeline behavior, power modes, data and instruction cache models, and bus interfaces. For custom HW the used RT component library, and for communication resources scheduling policies for shared media have to be specified.

3.3. Estimation & Model Generation

In order to allow a fast simulation and estimation, we create annotated C/C++/SystemC code. This annotated code contains information about the timing and power of each component. Power and timing information for each of them is obtained using existing and sophisticated tools as mentioned in Section 2.2.

As depicted in Section 2 a realistic system consists of components of different type e. g., custom hard- and software as well as IP-components, like communication infrastructure. Each component is estimated individually, using an appropriate tool. Based on the estimation an augmented version of the component is created, containing a power and timing model of the component.


Behavior Model
T0
S0
S1
T1
T2
parallel application description
Architecture Model
PPC
HW0
Mem
HW1
architecture/
platform description
mapping
Custom HW/SW Components
behavior
protocol
(shared) data
high-level Synthesis or software cross-compilation
power/timing estimation and back-annotation
cutsom hardware & software estimation
Platform dependent Behavior Model
virtual platform generation
timing & power aware executable TLM model
target arch. proper-ties
µP/µC
Bus IF
Bus
RAM
Bus Arbiter
timing & power aware executable RTL model
virtual platform gener-ation
IP Components
TLM
component library:
- initiator, target
- memory
- router
timing & power properties
IP & virtual component models (with power & timing annotations)
Abstract task (C/C++)
Constraints + Component library
Task stimuli
Control and data flow graph
Behavioral Synthesis
- Scheduling
- Allocation
- Binding
- Module selection
- Controller synthesis
- RTL floorplanning
RT-level power characterization
Power model library
Basic block identification & time/power annotation
Synthesizable RT-level Verilog
Timing and power annotated HW module
C Source code
Target Processor Library
LLVM-Based Front-end (Static Modeling)
LLVM Intermediate Representation
Static Model
Instrumentation
Instrumented LLVM code
Tracing, Back-end and Postprocessing
Back-annotated Source Code
Estimates and Analysis Reports

(a) Example of an Executable Specification and a Platform Mapping (b) HW estimation flow (c) SW estimation flow

Figure 2: The proposed virtual prototyping approach with timing and power back-annotation flows

From these annotated components a virtual prototype is generated. This prototype is used to estimate the power and timing of the overall system. In the next four paragraphs we describe these steps in more detail.

Hardware/Software task separation. Depending on the user-defined mapping, each behavior of the parallel application description is estimated with an appropriate technique. The tools, used for HW & SW estimation and characterization, perform a simulation-based estimation. Thus, each component must be simulated and characterized individually. The system is split into individual components. The surrounding system serves as testbench/test environment during the simulation. This way we can simulate each component individually and still obtain estimates, which correspond to the behavior of the overall system.

Custom Hardware estimation. Timing and power estimation of application specific hardware designs can be done at nearly every level of abstraction from transistor-up to behavioral level at ESL. Since we address behavioral tasks that are mapped to hardware and are meant to be implemented in custom ASIC hardware, we only address behavioral-level estimation here [65].

To consider the challenges of ASIC power-modeling mentioned in Section 2 we combine synthesis with cycle-accurate simulation at RT-level and a subsequent phase of basic-block identification and power/timing annotation. Although extensive power estimation at RT-level is very time consuming and thus not applicable in HW/SW co-simulation, it can be used as characterization approach for higher level estimation. This is why we apply lower-level estimation provided by the OFFIS PowerOpt tool to a small but typical testbench and derive cycle-averaged power estimates. These power values will then be further abstracted to basic block level and annotated to the internal control- and dataflow graph-representation (CDFG, see Figure 2(b)). We differentiate between dynamic and static power as well as its source (e. g., functional units, controller, or clock tree). Leakage power at RT-level is nearly independent on data pattern [66] (variation of 15 %) and thus it mainly depends on elapsed time whereas dynamic power depends on the testbench stimuli. In addition the delay is annotated to each basic block which is fixed and can be statically derived from the cycle count within the scheduled CDFG and the frequency. In the last step we export the CDFG to a timing and power annotated SystemC module. The overall HW-estimation flow is presented in Figure 2(b).

The presented custom hardware timing and power estimation technique supports the creation of virtual prototypes for embedded full-custom hardware modules. Based on the automatically generated cycle-accurate functional description at register transfer level a characterization of the module is performed and a high-level, C++-based virtual prototype is generated. It is augmented with RT level accurate power and timing information. First experiments on data intensive hardware accelerators show a fast and accurate estimation of power properties with a total error of about 3.6% and a speed-up of approximately 516 compared to an RT-level estimation, while obtaining cycle accurate timing information [67]. These properties support early design space exploration.

Software estimation. At the intermediate level of the COMPLEX tool chain, and integrated with software generation, hardware model characterization and design space exploration, the detailed software estimation tool set SWAT (*SoftWare Analysis Toolset*) plays a key role. Its main goal is to provide more accurate and detailed performance and energy estimates with respect to those obtained using higher-level and more abstract models [68].

The purpose of SWAT is to provide methodology and tools to estimate performance and energy requirements of embedded software, conveying the advantages of ISS approaches and dynamic approaches, and mitigating their disadvantages. The SWAT approach is based on a statistical analysis and characterization of an intermediate assembly-level code representation, instrumentation and execution on the host machine. The most significant novelties of the proposed methodology are:

- The methodology does not require to execute an ISS, even in those cases where the software is required to interface to the external world, e.g. to hardware peripherals on the target SoC. Suitably designed stubs can be developed and seamlessly integrated with power FSMs models.
- The performance and energy consumption models of the target microprocessor can be derived in almost completely automatic way by means of a specific set of tools that are part of SWAT. The accuracy of such models depends on the accuracy of the basic figures available for the target core.
- The intermediate code (LLVM) is very close to the target assembly code, but still sufficiently abstract to allow for a source-level analysis. The analysis performed on the intermediate code, thus, combines efficiency, accuracy and flexibility.
- Execution on the host machine does not require the entire hardware of the target system, reducing costs and times of the analysis.

The SWAT modeling approach can be split into three main phases:

1. *Target machine modeling.* Timing and energy consumption characteristics of the target architecture need to be modeled in a suitable mathematical form to feed the remaining phases of the flow. The first step, thus, consists in building this instruction-set model. This is done only once for each target platform.
2. *Source code modeling.* The second modeling phase is aimed at decoupling the influence of the static structure of the source code (i. e. its semantics) both from the specific target architecture and, most notably, from the specific input run-time data.
3. *Dynamic behavior modeling.* The dynamic behavior is accounted by performing a basic-block profiling at intermediate-representation level (i. e. LLVM code). Such a profiling can be accomplished by executing the code on a host machine, possibly resorting to some support libraries that model hardware devices.
4. *Post-processing.* A final phase combines all the models and derives timing and energy estimates of a specific (or a set of) execution. Beyond a wide range of statistics concerning static and dynamic properties of the application, this last phase back-annotates performance figures directly on the original source code. This is especially valuable to support the application developer in the subsequent optimization phase.

Each phase has been studied accurately and sound mathematical models have been built as a foundation. A simplified view of the resulting estimation flow is depicted in Figure 2(c).

The methodology has been applied to several benchmarks and the estimates compared with the most accurate available figures, i. e. those obtained with a target-specific, power-enabled instruction-set simulator [69]. The absolute estimation errors obtained on a set of several benchmarks ranges from less than 2% up to to 13%, with an average of 6%. The SWAT estimation flow has been proven to be much more efficient than ISS-based analysis, namely more than 400 times faster[1]. This speed-up, combined with a satisfactory accuracy, allows to integrate the SWAT methodology within a design-space exploration framework, in particular the MOST DSE engine.

Pre-existing IP & virtual component models. Despite custom HW and SW, preexisting or third-party components must be considered. Models for communication infrastructure like buses are provided by different vendors and typically are provided as parameterizable soft-macros, allowing an adaption to the system to be build. The macros already contain timing information but typically no information about power. Thus, communication power is estimated based on the TLM-2.0 transport calls. Calculation accounts for the size of transferred data, type of access, as well as duration of the communication. Interruptions and re-arbitration events are also considered and must be delivered by the communication model.

General IP components delivered by third-party vendors cannot be estimated like custom HW and SW components. System-level representatives of these IP's are typically provided as black-box executable models (e. g.

[1]These results refer to a ReISC III core with 1.0 V power supply and operating at 50 MHz.

API to a compiled object-file). These black-box modules typically contain timing but no power information. In order to obtain at least approximated power values a simple wrapper or monitor is used, which monitors the components in- and output. This information is used to control a power state machine (PSM) [70] inside the monitor. Power states and power values of the PSM are either obtained from the component's data-sheet or estimated manually.

Latest results on an abstraction methodology for generating time- and power-annotated TLM models from synthesizable RTL descriptions can be found in [71]. The proposed techniques allow the integration of existing RTL IP components into virtual platforms for early software development and platform design, configuration, and exploration. With the proposed approach, IP models can be natively integrated into SystemC TLM-2.0 platforms and executed 10-1000 times faster compared to state-of-the-art RTL simulators. The abstraction methodology guarantees preservation of the behavior and timing of the RTL models. Target technology dependent power properties of IP components are represented as power state-machines and integrated into the abstracted TLM models. The experimental results show a relative error less than 10% of the abstracted model's power consumption compared to state-of-the-art RTL power simulators [72].

Virtual system generation. During generation of the virtual system annotated source from ⓕ as well as the selected models from ⓘ are combined to a virtual prototype. The example in Figure 3(a) shows the virtual platform model obtained from the mapping specified in Figure 2(a). The timing and power annotated execution-models are integrated with timing and power characterized platform models. In the example in Figure 3(a) these platform models are: a TLM-2.0 router with bus protocol and power model, and a system memory model. For the integration of the annotated task behavior with the TLM communication network we provide communication interface (IF) templates. These interfaces translate the function calls of the active tasks into TLM transaction containers. For passive tasks we synthesize a memory interface which decouples the TLM transactions from the activation of the behavior. That means the transaction is stored in the memory completely, before the passive task is activated. These interface templates are timing and power characterized for the chosen platform.

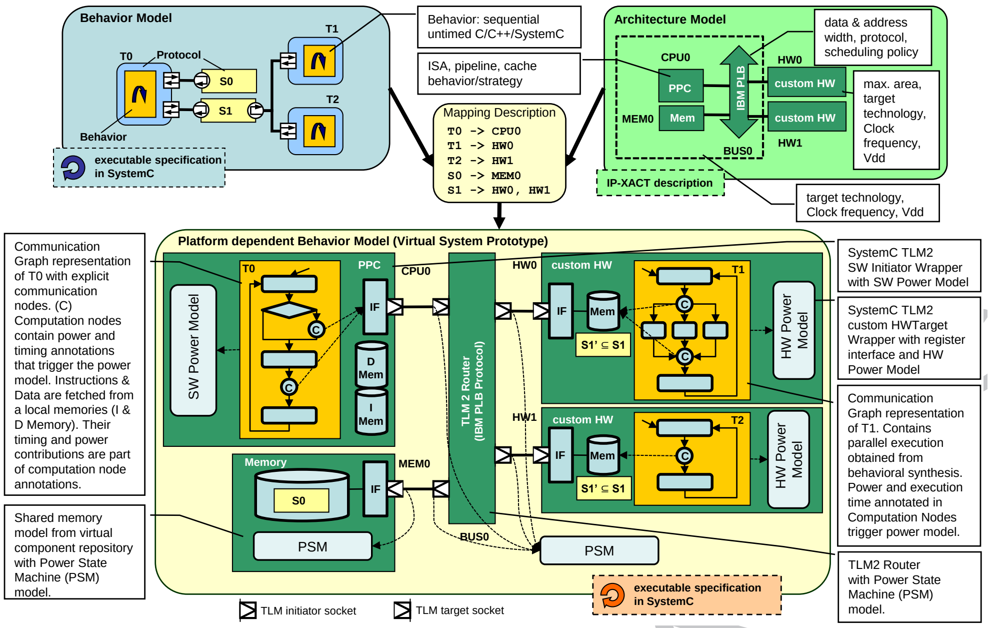
3.4. Simulation

Pre-optimized power controller. The pre-optimized power controller implements a framework for dynamic adaptation to changing context and transparent optimization of platform resource usage [58]. It follows a distributed and hierarchical approach. On the one hand, a Global Resource Manager (GRM) is loaded on the host processor of the platform. It is a software task running in parallel with the application. It is a middleware providing a bridge between the application, the user and the platform. It conforms to practices of each Local Resource Manager (LRM) in each platform IP core (e.g., HW block or SW processor). It is used to adapt both platform and application at run time and to find global and optimal trade-offs in application mapping based on a given optimization goal. On the other hand, each IP core can execute its own resource management without any restriction, through an LRM. Such an LRM encapsulates the local policies and mechanisms used to initiate, monitor and control computation on its IP core.

In contrast to the collaboration between the GRM and the LRMs, the GRM collaboration with application and user is visible to the application developer and is performed as follows. First, the QoS requirements and the optimization goal are set by the user. The goal is then translated into an abstract and mathematical function, called utility function (e.g., performance, power consumption, battery life, QoS weighted combination of them). Then, at run time, the GRM manages and optimizes the application mapping taking into account the possible application configurations explored at design time, the platform resources currently available, the QoS requirements, and the utility function.

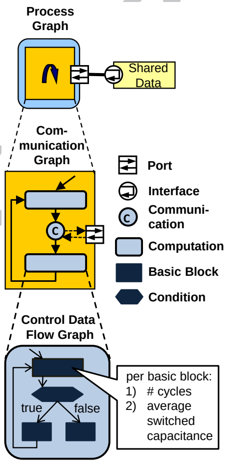
Timing & power aware executable virtual system prototype in SystemC. During system execution the annotated timing and power information is collected. Depending on the workload model different execution paths, leading to different timing and power values, are possible. After simulation, the collected information can be illustrated in a power-over-time diagram or can be used for a power-breakdown. Our annotations can be traced at different levels of granularity to allow a user-defined trade-off between performance and accuracy. Figure 3(b) depicts the different timing and power evaluation levels.

On the most abstract level we only consider analysis on *task granularity*. This can be easily performed between active and passive tasks with blocking communication relation. Execution time and power of the passive task can simply be inlined and accumulated with time



Behavior Model
Behavior: sequential untimed C/C++/SystemC
ISA, pipeline, cache behavior/strategy
Architecture Model
data & address width, protocol, scheduling policy
max. area, target technology, Clock frequency, Vdd
Mapping Description
T0 -> CPU0
T1 -> HW0
T2 -> HW1
S0 -> MEM0
S1 -> HW0, HW1
IP-XACT description
target technology, Clock frequency, Vdd
executable specification in SystemC
Platform dependent Behavior Model (Virtual System Prototype)
Communication Graph representation of T0 with explicit communication nodes. (C) Computation nodes contain power and timing annotations that trigger the power model. Instructions & Data are fetched from a local memories (I & D Memory). Their timing and power contributions are part of computation node annotations.
Shared memory model from virtual component repository with Power State Machine (PSM) model.
SystemC TLM2 SW Initiator Wrapper with SW Power Model
SystemC TLM2 custom HWTarget Wrapper with register interface and HW Power Model
Communication Graph representation of T1. Contains parallel execution behavioral synthesis. Power and execution time annotated in Computation Nodes trigger power model.
TLM2 Router with Power State Machine (PSM) model.
SW Power Model
HW Power Model
TLM 2 Router (IBM PLB Protocol)
PSM
BUS0
TLM initiator socket
TLM target socket

(a) Virtual Platform generated from example shown in Figure 2(a)



(b) Annotations

Figure 3: Example of a Virtual Platform Model

and power of the active task. The next level of granularity works on *communication granularity*. Power and timing of computation nodes in the communication graph are accumulated and only traced at the time points of communication. For a deeper analysis of the timing and power behavior traces on *basic block granularity* of a CDFG is also possible.

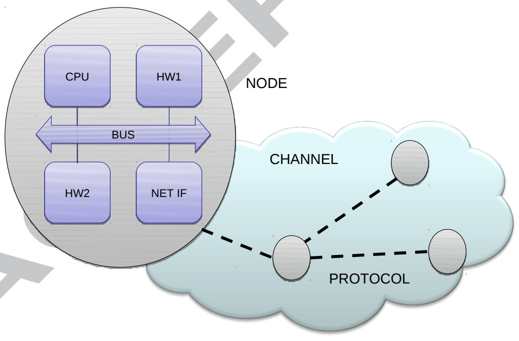


Figure 4: Network View: the virtual platform of a networked embedded system is simulated together with a model of the network.

A possible new simulation view provided by the COMPLEX project is the so-called *Network View* depicted in Figure 4 and enabled by the SystemC Network Simulation Library (SCNSL) [73]. It is an extension of SystemC to allow modeling packet-based networks such as wireless networks, Ethernet, and field bus. It supports the simulation of packet transmission, reception, contention on the channel and wireless path loss.

The advantages of SCNSL are:

- simplicity: a single language/tool, i. e. SystemC, is used to model both the system (i. e. CPU, memory, peripherals) and the communication network;
- efficiency: faster simulations can be performed since no external network simulator is required;
- re-use of SystemC IP blocks
- scalability: support of different abstraction levels in the design description
- openness: several tools available for SystemC can be exploited seamlessly
- extensibility: the use of standard SystemC and the source code availability guarantee the extensibility of the library to meet design-specific constraints

According to Figure 4, the traditional virtual platform, made of CPU, HW blocks and bus, can be extended by wrapping it into a *network node* exchanging packets with other nodes through a *channel* by using well-known *protocols* such as IEEE 802.15.4. All these elements are provided by SCNSL as traditional SystemC blocks. SCNSL also allows improving timing and power analysis by introducing the effect of communications which have a direct impact on timing behavior and power consumption of the system under design [74].

3.5. *Exploration & Optimization*

The Exploration and Optimization phase creates a feedback loop between the performance estimation part (including MDA Design Entry, Executable Specification, Estimation & Model Generation and Simulation phases) and the parameters configuration of the target system. In particular, the loop is closed by acquiring the system simulation traces to be post processed before being presented either to the user or to an automatic framework for exploration and optimization. Then, according to the information gathered by the previous simulation, a new HW/SW system configuration will be selected within the design space. In the next paragraph we describe these steps in more detail.

Simulation trace. Our proposed simulation and tracing framework supports different granularities for each task. Moreover, it is possible to define certain regions of interest in the sequential behavior of a task that can be investigated down to basic block granularity.

Trace analysis. The simulation and analysis of timing and power traces allows an evaluation of the chosen application mapping, the performance of the architecture and the effects of the synthesis constraints. Different design configurations or iterations with adjusted mapping, platform composition and constraints allow multi-objective design-space exploration.

Visualization & reporting. The visualization and reporting step in the proposed framework has in charge the presentation of the results coming from the trace analysis phase into a readable format for the designer. This phase is necessary to have the designer driving the exploration loop. Moreover, this phase includes also the visualization of the analysis coming from the exploration loop done by using the MOST tool.

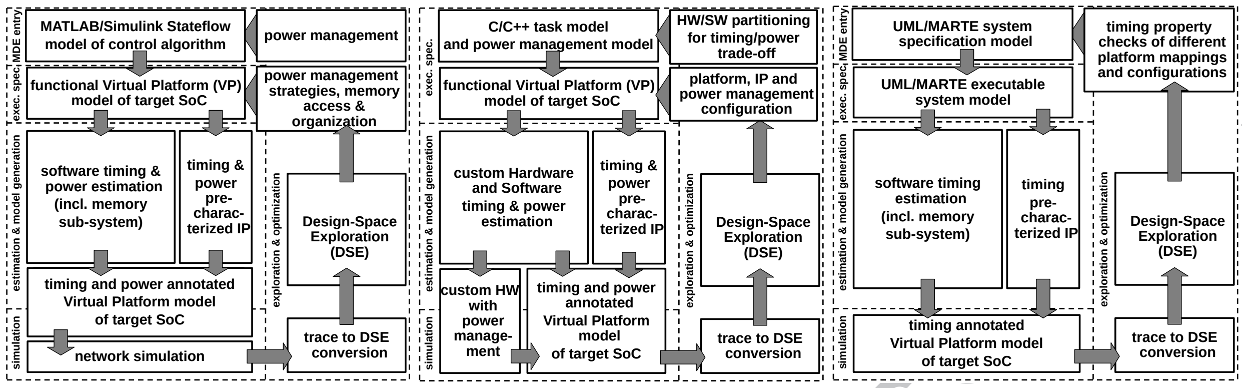
Exploration & optimization. Starting from the definition of the design space, the exploration and optimization step iteratively generates an instance of the design space based on the knowledge acquired by the post-processing of the simulation traces of previous selected configurations. The exploration phase is a step in the design flow that is needed for surfing the design space (changing the system parameters) in order to find the optimal system configurations among all the possible alternatives that are part of the design space. Moreover, the design space exploration loop is also used to determine some knowledge about the system parameters (such as the main effects, interaction effects) and design space (such as, configuration distribution with respect to the system performance). This phase can be done by using a user centric DSE or an automatic DSE phase. The goal in using an automatic design space exploration and optimization tool is in the fact that it should be able to automatically interact with system models in order to avoid the intervention of the designer for the DSE phase (except for the analysis of the results) once the target problem is formally defined. The automatic design space exploration tool used within the project is called MOST, and the interfaces used to interact with the other components of the design flow are those presented in [54]. On the other side, the usage of a user centric DSE flow is needed when a detailed analysis of the system behavior is necessary (e.g. trace analysis or time behavior), once the problem cannot be formally defined or it is not easy to be defined, or when the automatic modification of the parameters on the system model is not possible or requires a larger modeling effort.

Design space definition. The design space is defined by the list of tunable parameters available on the HW/SW platform. Moreover, it includes the set of possible values of each parameters and the rules defining some cuts within the design space eventually due to interferences between parameters. The design space definition represents the degrees of freedom that the designer or an automatic tool can have for tuning the HW/SW platform.

Design space instance parameters. A design space instance is a valid configuration of parameters within the design space defined before selected or by the designer or by an automatic tool. It is composed of a value for each parameter of the design space, ranging within the set of available levels. Those values will be used to fill the right parameter values in several stages of the design-flow (see Figure 1). In the project, the list of parameters ranges at the MDA Design Entry and Executable Specification levels from functional reimplementation to mapping of HW/SW tasks and IP selection, while at Estimation and Model Generation level from IP and memories configuration to selection of embedded software optimizations and run-time management strategies.

4. Use-Cases aware design flow customization

The proposed reference design flow can be customized for different industrial use-cases (UC). Figure 5 visualizes three different configurations of the COMPLEX reference design flow, as shown in Figure 1. The following Sections will briefly describe the different use-cases and the application of the proposed flow.


MDE entry
exec. spec.
estimation & model generation
simulation
exploration & optimization
MATLAB/Simulink Stateflow model of control algorithm
power management
functional Virtual Platform (VP) model of target SoC
power management strategies, memory access & organization
software timing & power estimation (incl. memory sub-system)
timing & power pre-charac-terized IP
Design-Space Exploration (DSE)
timing and power annotated Virtual Platform model of target SoC
network simulation
trace to DSE conversion
C/C++ task model and power management model
HW/SW partitioning for timing/power trade-off
platform, IP and power management configuration
custom Hardware and Software timing & power estimation
custom HW with power manage-ment
timing and power annotated Virtual Platform model of target SoC
UML/MARTE system specification model
timing property checks of different platform mappings and configurations
UML/MARTE executable system model
software timing estimation (incl. memory sub-system)
timing pre-charac-terized IP
timing annotated Virtual Platform model of target SoC

(a) UC1: Networked control system (b) UC2: Battery powered multi-core system (c) UC3: Real-time system

Figure 5: Different configurations of the COMPLEX reference design flow

4.1. Use-Case 1 – Networked embedded system

Industrial requirements on the COMPLEX flow are an optimal software mapping solution for real-time, low power applications based on pre-existent hardware low power techniques, methods to build a power aware HW virtual platform, and the possibility to easily integrate the COMPLEX flow into an industrial state-of-the art design flow.

The Application comes from the health care domain. It is a virtual machine oriented to data processing in body sensor networks. A node of this application is able to perform some computations (such as max, min, median, etc.) based on collected data sets from sensors. The parameters of these computations (called ”features”) such as sampling rate, window, shift of data set, etc. can be tuned and the features can be activated or deactivated depending on the application demands. Parameter tuning and activation/deactivation of features can be done at run time. One prominent application of this virtual machine will be to detect body movements, for example to monitor the health state of an elderly patient.

The Platform is the ReISC SoC, a project developed at STMicroelectronics, with a 32-bit RISC core operating at up to 50 MHz, embedded memories, and an extensive range of enhanced I/Os and peripherals. A comprehensive set of power-saving mode allows the design of low-power applications. The CPU of the SoC executes the application SW, operating system (FreeRTOS), drivers, and interrupt service routines. The use-case consists of a wireless sensor network (WSN) scenario, with multiple ReISC SoCs connected via radio. All nodes and the radio channel are simulated using the SCNSL network simulator.

Application mapping starts from a Stateflow description of the application, that will be translated in SystemC using HFSuite and ported to the ReISC SoC running FreeRTOS [75].

Goal of the evaluation is to reach a configuration with minimum power consumption respecting real-time constraint, by acting to different power management strategies (clock gating, power island) and memory access optimization. Software power management policies are driven by traffic congestion reproduced by using the network simulator [74].

4.2. Use-Case 2 – Battery powered multi-core system

Industrial requirements are assisted HW/SW separation with automatic generation of HW wrappers, efficient handling of communications between automatically generated modules of the design, fast simulation, possibility to observe impact of platform and application parameters on power and timing.

The application provides a modular solution for audio-driven activity monitoring in the context of a battery-powered surveillance application. Its set of processing modules can be dynamically activated according to the evolving environmental conditions.

The hardware platform is a homogeneous MPSoC platform, with a main processing unit (PU) acting as host, controlling different slave/co-processing units that can be either software processors or custom hardware accelerators. For power management different power islands (one power island for each PU) exist.

The application mapping allows the possibility to select the power-island state of each PU, where the host and audio monitoring PUs are always active. The Central and Global Resource managers handle application and platform configurations at run-time. The quality of service of the system is configurable from 1 to 4 microphones with a variable set of features.

Goal of the evaluation is to demonstrate the efficiency of the COMPLEX flow, from SystemC specifications to DSE optimization. Moreover, this evaluation shall show the complementary nature of design-time and run-time optimizations. The demonstrator aims at highlighting:

- relevance of HW/SW separation decisions and efficiency of generated code
- performance of the virtual platform: simulation speed vs. accuracy of estimates
- flexibility of the Design Space Exploration tool: multi-objectives optimization
- efficiency of the Global Resource Manager: quality of service adaptation to maximize battery lifetime

4.3. Use-Case 3 – Model-based space application

Industrial requirements are support of model-driven design combined with design-space exploration of embedded HW/SW systems and evaluation of the COMPLEX design flow and tools, with a special focus on the model-based design tools and transformation engines.

The application is an object survey, tracking and imaging system in the context of the Space Situational Awareness (SSA) in a star tracer satellite system. For evaluation in COMPLEX a simplified version of the real system has been chosen, that consists of the following main functional blocks:

- Image processing: image acquisition, image transformations and image clutter filtering
- Object survey, tracking, and hazard analysis
- GPS SW receiver: signal acquisition, FFT, correlation, channel tracking, as well as navigation and position measurements

The application is represented as UML/MARTE Platform Independent Model (PIM).

The platform consists of three processing boards, each of them is equipped with a processor from the ARM family. Boards are interconnected via serial buses. The GPS SW module can access HW accelerators to perform computation intensive tasks like an FFT. The execution platform is represented as UML/MARTE Platform Description Model (PDM).

The application mapping is represented as UML/-MARTE Platform Specific Model (PSM). It defines the allocation space and design space constraints. The PIM, PDM, and PSM are used to generate an executable model for design-space exploration through the following transformations:

- UML/MARTE to executable application model
- UML/MARTE to DSE loop configuration and allocation space definition
- UML/MARTE to executable environment/testbench model

In a second step, after design-space exploration, an executable Virtual Platform can be generated through UML/MARTE to IP-XACT and IP-XACT to SystemC transformation steps.

Specifically, the UML/MARTE COMPLEX flow [60] developed for the use case 3 has demonstrated that additional improvements to the general exploration flow described in the previous section are possible. Specifically, the UML/MARTE COMPLEX flow removes the MDA entry from the DSE loop, and avoids the recompilation of the executable performance model for each iteration. This way, a significant speed-up of the DSE loop is achieved.

Goal of the evaluation is the assessment of high-level functional analysis combined with performance and power estimation, under usage of techniques for exploration of the optimal architecture based on UML/-MARTE models. Virtual platform reference implementations will be used for comparison of metrics from different timing and power estimation techniques, i.e. ISS simulator versus high-level estimation techniques.

5. Conclusion

In this paper we presented the COMPLEX reference framework and design flow allowing rapid virtual prototyping of heterogeneous embedded HW/SW systems under consideration of timing and power aspects. The presented approach enables a UML/MARTE design entry with automatic generation of an executable SystemC model. The extra-functional estimation flow considers custom hard- and software as well as third party IP components. For each of these different types we use state-of-the-art commercial tools, inducing satisfying estimates for individual components. A generated virtual prototype allows a fast and unitary simulation of the complete system's functional and extra-functional properties, which helps the designer to identify and eliminate performance bottlenecks and power peaks. Thus, a fast and early trade-off between different design alternatives becomes feasible. To support an efficient exploration we have extended our flow by a feedback loop, allowing guided iterations through different configurations, mappings, power management policies, etc. Based on the industrial use-cases [60, 58, 74] we are convinced that the overall framework will integrate existing approaches and point-tools, into a common and unified flow for timing and power aware rapid prototyping.

Acknowledgment

The authors would like to thank all partners involved in the COMPLEX FP7 European Integrated Project [6], funded by the European Commission under Grant Agreement 247999.

References

[1] L. Zhong, S. Ravi, A. Raghunathan, N. Jha, RTL-Aware Cycle-Accurate Functional Power Estimation, Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on 25 (10) (2006) 2103 –2117. doi:10.1109/TCAD.2005.859504.

[2] B. Sander, J. Schnerr, O. Bringmann, ESL power analysis of embedded processors for temperature and reliability estimations, in: Proceedings of the 7th IEEE/ACM international conference on Hardware/software codesign and system synthesis, CODES+ISSS '09, ACM, New York, NY, USA, 2009, pp. 239–248. doi:10.1145/1629435.1629469.

[3] N. Bansal, K. Lahiri, A. Raghunathan, S. T. Chakradhar, Power Monitors: A Framework for System-Level Power Estimation Using Heterogeneous Power Models, in: Proceedings of the 18th International Conference on VLSI Design held jointly with 4th International Conference on Embedded Systems Design, VLSID '05, IEEE Computer Society, Washington, DC, USA, 2005, pp. 579–585. doi:10.1109/ICVD.2005.138.

[4] N. Dhanwada, I.-C. Lin, V. Narayanan, A power estimation methodology for SystemC transaction level models, in: Proceedings of the 3rd IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis, CODES+ISSS '05, ACM, New York, NY, USA, 2005, pp. 142–147. doi:10.1145/1084834.1084874.

[5] H. Lebreton, P. Vivet, Power Modeling in SystemC at Transaction Level, Application to a DVFS Architecture, in: Symposium on VLSI, 2008. ISVLSI '08. IEEE Computer Society Annual, 2008, pp. 463 –466. doi:10.1109/ISVLSI.2008.71.

[6] The COMPLEX project (FP7-247999), COdesign and power Management in PLatform-based design space EXploration.
URL http://complex.offis.de

[7] A. Sangiovanni-Vincentelli, Is a Unified Methodology for System-Level Design Possible?, IEEE Design and Test of Computers 25 (4) (2008) 346–357.

[8] G. Palermo, C. Brandolese, F. Ferrero, F. Herrera, G. Schomaker, C. Brunzema, K. Grüttner, K. Hylla, B. Vanthournout, D. Quaglia, L. Lavagno, M. Poncino, E. Vaumorin, C. Couvreur, S. A. Butt, Definition of application, stimuli and platform specification, and definition of tool interfaces, Tech. Rep. COMPLEX/PoliMi/R/D1.2.1/1.1, COMPLEX project deliverable (Oct. 2010).
URL http://complex.offis.de/docs/8

[9] D. Densmore, R. Passerone, A. Sangiovanni-Vincentelli, A Platform-Based Taxonomy for ESL Design, IEEE Design and Test of Computers 23 (5) (2006) 359–374.

[10] D. C. Schmidt, Model-driven engineering, Computer – IEEE Computer Society 39 (2) (2006) 25.

[11] A. Sangiovanni-Vincentelli, G. Martin, Platform-Based Design and Software Design Methodology for Embedded Systems, IEEE Design and Test of Computers 18 (6) (2001) 23–33.

[12] A. Sangiovanni-Vincentelli, Quo Vadis SLD: Reasoning about Trends and Challenges of System-Level Design, Proceedings of the IEEE 95 (3) (2007) 467–506.

[13] Y. Vanderperren, W. Dehaene, A Model Driven Development Process for Low Power SoC Using UML, Springer, 2005.

[14] A. Lakshminarayana, S. Ahuja, S. Shukla, Coprocessor design space exploration using high level synthesis, in: Quality Electronic Design (ISQED), 2010 11th International Symposium on, 2010, pp. 879–884. doi:10.1109/ISQED.2010.5450474.

[15] Synopsys, Synphony High-Level Synthesis, http://www.synopsys.com/Systems/BlockDesign/HLS/Pages/default.aspx.

[16] Calypto Design Systems, Catapult, http://calypto.com/en/products/catapult/overview.

[17] Cadence, C-to-silicon compiler, http://www.cadence.com/products/sd/silicon_compiler/pages/default.aspx.

[18] Forte Design Systems, Cynthesizer, http://www.forteds.com/.

[19] M. Holzer, M. Rupp, Static Estimation of Execution Times for Hardware Accelerators in System-on-Chips, in: System-on-Chip, 2005. Proceedings. 2005 International Symposium on, 2005, pp. 62–65. doi:10.1109/ISSOC.2005.1595645.

[20] M. Sangeetha, J. RajaPaul Perinbam, Revathy, Hardware Estimation and Synthesis for a Codesign System, in: Signal Processing, Communications and Networking, 2007. ICSCN '07. International Conference on, 2007, pp. 189–194. doi:10.1109/ICSCN.2007.350728.

[21] M. B. Abdelhalim, S. Habib, Fast FPGA-based area and latency estimation for a novel hardware/software partitioning scheme, in: Electrical and Computer Engineering, 2008. CCECE 2008. Canadian Conference on, 2008, pp. 000775–000780. doi:10.1109/CCECE.2008.4564641.

[22] B. Dwivedi, A. Kejariwal, M. Balakrishnan, A. Kumar, Rapid Resource-Constrained Hardware Performance Estimation, in: Rapid System Prototyping, 2006. Seventeenth IEEE International Workshop on, 2006, pp. 40–46. doi:10.1109/RSP.2006.33.

[23] L. Deng, K. Sobti, C. Chakrabarti, Accurate models for estimating area and power of FPGA implementations, in: Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on, 2008, pp. 1417–1420. doi:10.1109/ICASSP.2008.4517885.

[24] R. Meeuws, Y. Yankova, K. Bertels, G. Gaydadjiev, S. Vassiliadis, A. Heterogeneous, S. Development, A quantitative prediction model for hardware/software partitioning, in: in Proceedings of 17th International Conference on Field Programmable Logic and Applications (FPL07, 2007, p. 5.

[25] P. Bjuréus, M. Millberg, A. Jantsch, FPGA resource and timing estimation from Matlab execution traces, in: Proceedings of the tenth international symposium on Hardware/software codesign, CODES '02, ACM, New York, NY, USA, 2002, pp. 31–36. doi:10.1145/774789.774797.
URL http://doi.acm.org/10.1145/774789.774797

[26] D. Brooks, V. Tiwari, M. Martonosi, Wattch: A framework for architectural-level power analysis and optimizations, In Proceedings of the 27th Annual International Symposium on Computer Architecture (2000) 83–94.

[27] W. Ye, N. Vijaykrishnan, M. Kandemir, M. J. Irwin, The design and use of simplepower: a cycle-accurate energy estimation tool, in: Proceedings of the 37th Annual Design Automation Conference, DAC '00, ACM, New York, NY, USA, 2000, pp. 340–345. doi:10.1145/337292.337436.
URL http://doi.acm.org/10.1145/337292.337436

[28] A. Sinha, A. Chandrakasan, JouleTrack - A Web Based Tool for Software Energy Profiling, Proceedings on Design Automation

Conference (2001) 220–225.
[29] T. K. Tan, A. Raghunathan, G. Lakshminarayana, N. Jha, High-level software energy macromodeling, Proceedings on Design Automation Conference (2001) 605–610.
[30] P. P. Sotiriadis, A. P. Chandrakasan, A bus energy model for deep submicron technology, IEEE Trans. Very Large Scale Integr. Syst. 10 (3) (2002) 341–350. `doi:10.1109/TVLSI.2002.1043337`.
URL `http://dx.doi.org/10.1109/TVLSI.2002.1043337`
[31] M. B. Kamble, K. Ghose, Analytical energy dissipation models for low-power caches, in: Proceedings of the 1997 international symposium on Low power electronics and design, ISLPED '97, ACM, New York, NY, USA, 1997, pp. 143–148. `doi:10.1145/263272.263310`.
URL `http://doi.acm.org/10.1145/263272.263310`
[32] T. Givargis, F. Vahid, J. Henkel, S. Member, Evaluating Power Consumption of Parameterized Cache and Bus Architectures in System-on-a-Chip Designs, IEEE Trans. VLSI Systems 9 (2001) 500–508.
[33] R. Mehra, J. Rabaey, Behavioral Level Power Estimation and Exploration, in: in Proc. Int. Wkshp. Low Power Design, 1994, pp. 197–202.
[34] L. Kruse, E. Schmidt, G. Jochens, A. Stammermann, A. Schulz, E. Macii, W. Nebel, Estimation of lower and upper bounds on the power consumption from scheduled data flow graphs, IEEE Trans. Very Large Scale Integr. Syst. 9 (1) (2001) 3–15. `doi:10.1109/92.920813`.
URL `http://dx.doi.org/10.1109/92.920813`
[35] L. Zhong, S. Ravi, A. Raghunathan, N. K. Jha, Power estimation for cycle-accurate functional descriptions of hardware, in: Proceedings of the 2004 IEEE/ACM International conference on Computer-aided design, ICCAD '04, IEEE Computer Society, Washington, DC, USA, 2004, pp. 668–675. `doi:10.1109/ICCAD.2004.1382659`.
URL `http://dx.doi.org/10.1109/ICCAD.2004.1382659`
[36] DVM, `http://aldec.com`.
[37] VHDLC, `http://ostatic.com`.
[38] FreeHDL-V2CC, `http://linux.die.net/man/1/freehdl-v2cc`.
[39] W. Snyder, P. Wasson, D. Galbi, Verilator - Convert Verilog code to C++/SystemC, `http://www.veripool.org/wiki/verilator`.
[40] W. Stoye, D. Greaves, N. Richards, J. Green, Using RTL-to-C++ Translation for Large SoC Concurrent Engineering: A Case Study, IEEE Electronics Systems and Software 1 (1) (2003) 20–25.
[41] Carbon Model Studio, `http://carbondesignsystems.com/`.
[42] R. Görgen, J.-H. Oetjens, W. Nebel, Transformation of Event-Driven HDL Blocks for Native Integration into Time-Driven System Models, in: Proceedings of the Forum on specification and Design Languages (FDL'2012), 2012.
[43] F. Bellard, Qemu, a fast and portable dynamic translator, in: Proc. of the USENIX Annual Technical Conference, 2005, pp. 41–46.
[44] L. Benini, R. Hodgson, P. Siegel, System-level power estimation and optimization, in: International Symposium on Low Power Electronics and Design, ISLPED'98, ACM, Monterey, CA, USA, 1998, pp. 173–178.
[45] C. Walravens, Y. Vanderperren, W. Dehaene, ActivaSC: A highly efficient and non-intrusive extension for activity-based analysis of SystemC models, in: 46th ACM/IEEE Design Automation Conference (DAC'2009), 2009, pp. 172–177.
[46] M. Giammarini, S. Orcioni, M. Conti, Powersim: Power Estimation with SystemC, in: Solutions on Embedded Systems, Vol. 81 of Lecture Notes in Electrical Engineering, 2011, pp. 285–300. `doi:10.1007/978-94-007-0638-5_20`.
[47] M. Streubühr, R. Rosales, R. Hasholzner, C. Haubelt, J. Teich, ESL Power and Performance Estimation for Heterogeneous MPSoCs Using SystemC, in: Forum on specification and Design Languages (FDL'2011), Oldenburg, Germany, 2011.
[48] Open Virtual Platforms (OVP) portal (2013).
URL `http://www.ovpworld.org/`
[49] S. McCanne, S. Floyd, NS Network Simulator – version 2, URL: http://www.isi.edu/nsnam/ns.
[50] C. Zhu, O. Yang, J. Aweya, M. Ouellette, D. Montuno, A comparison of active queue management algorithms using the OPNET Modeler, IEEE Communications Magazine 40 (6) (2002) 158–167.
[51] Varga, A. OMNET++, OMNET++. OMNET++ Manuals, URL: http://www.omnetpp.org.
[52] M. Chidester, A. George, Parallel simulation of chip-multiprocessor architectures, ACM Trans. Model. Comput. Simul. 12 (3) (2002) 176–200. `doi:10.1145/643114.643116`.
URL `http://doi.acm.org/10.1145/643114.643116`
[53] G. Palermo, C. Silvano, V. Zaccaria, ReSPIR: a response surface-based Pareto iterative refinement for application-specific design space exploration, Trans. Comp.-Aided Des. Integ. Cir. Sys. 28 (12) (2009) 1816–1829. `doi:10.1109/TCAD.2009.2028681`.
URL `http://dx.doi.org/10.1109/TCAD.2009.2028681`
[54] C. Silvano, W. Fornaciari, G. Palermo, V. Zaccaria, F. Castro, M. Martinez, S. Bocchio, R. Zafalon, P. Avasare, G. Vanmeerbeeck, C. Ykman-Couvreur, M. Wouters, C. Kavka, L. Onesti, A. Turco, U. Bondi, G. Marianik, H. Posadas, E. Villar, C. Wu, F. Dongrui, Z. Hao, T. Shibin, MULTICUBE: Multi-objective Design Space Exploration of Multi-core Architectures, in: 2010 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), 2010, pp. 488 –493.
[55] S. Bleuler, M. Laumanns, L. Thiele, E. Zitzler, PISA - A Platform and Programming Language Independent Interface for Search Algorithms (2003).
URL `citeseer.ist.psu.edu/bleuler03pisa.html`
[56] G. Mariani, G. Palermo, V. Zaccaria, C. Silvano, OSCAR: An Optimization Methodology Exploiting Spatial Correlation in Multicore Design Spaces, Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on 31 (5) (2012) 740–753. `doi:10.1109/TCAD.2011.2177457`.
[57] E. İpek, S. A. McKee, R. Caruana, B. R. de Supinski, M. Schulz, Efficiently exploring architectural design spaces via predictive modeling, Proceedings of the 12th international conference on Architectural support for programming languages and operating systems 40 (5) (2006) 195–206. `doi:http://doi.acm.org/10.1145/1168917.1168882`.
[58] C. Ykman-Couvreur, P. A. Hartmann, G. Palermo, F. Colas-Bigey, L. San, Run-time Resource Management based on Design Space Exploration, in: International Conference on Hardware/Software Codesign and System Synthesis, CODES+ISSS'2012, Tampere, FI, 2012.
[59] G. Mariani, V. Sima, G. Palermo, V. Zaccaria, C. Silvano, K. Bertels, Using multi-objective design space exploration to enable run-time resource management for reconfigurable architectures, in: Design, Automation Test in Europe Conference Exhibition (DATE), 2012, 2012, pp. 1379–1384. `doi:10.1109/DATE.2012.6176578`.
[60] F. Herrera, H. Posadas, P. Penil, E. Villar, F. Ferrero, R. Valencia, An MDD Methodology for Specification of Embed-

ded Systems and Automatic Generation of Fast Configurable and Executable Performance Models, in: International Conference on Hardware/Software Codesign and System Synthesis, CODES+ISSS'2012, Tampere, FI, 2012.

[61] F. Ferrero, R. Valencia, F. Herrera, E. Villar, L. Lavagno, D. Quaglia, System specification methodology using MARTE and Stateflow, Tech. Rep. COMPLEX/GMV/R/D2.1.1/1.1, COMPLEX project deliverable (Dec. 2010).
URL `http://complex.offis.de/docs/11`

[62] F. Herrera, P. Peñil, E. Villar, F. Ferrero, R. Valencia, L. Lavagno, D. Quaglia, SystemC generation tools from MARTE and Stateflow, Tech. Rep. COMPLEX/UC/P/D2.1.2/1.0, COMPLEX project deliverable (Jun. 2011).
URL `http://complex.offis.de/docs/21`

[63] K. Grüttner, C. Grabbe, F. Oppenheimer, W. Nebel, Object Oriented Design and Synthesis of Communication in Hardware-/Software Systems with OSSS, in: Proceedings of the SASIMI 2007, 2007.

[64] K. Grüttner, H. Andreas, P. A. Hartmann, A. Schallenberg, C. Brunzema, OSSS - A Library for Synthesisable System Level Models in SystemC$^{\text{TM}}$ (2008).
URL `http://www.system-synthesis.org`

[65] K. Hylla, P. González, P. Sánchez, F. Herrera, Final report on custom hardware estimation and model generation, Tech. Rep. COMPLEX/OFFIS/P/D2.4.2/1.0, COMPLEX project deliverable (Jan. 2012).
URL `http://complex.offis.de/docs/33`

[66] D. Helms, G. Ehmen, W. Nebel, Analysis and Modeling of Subthreshold Leakage of RT-Components under PTV and State Variation, Proceedings on International Symposium on Low Power Electronics and Design.

[67] K. Hylla, P. A. Hartmann, D. Helms, W. Nebel, Early Power & Timing Estimation of Custom Hardware Blocks based on Automatically Generated Combinatorial Macros, in: C. Haubelt, D. Timmermann (Eds.), MBMV, Institut für Angewandte Mikroelektronik und Datentechnik, Fakultät für Informatik und Elektrotechnik, Universität Rostock, 2013, pp. 147–158.

[68] C. Brandolese, G. Palermo, W. Fornaciari, H. Posadas, F. Herrera, P. Peñil, E. Villar, F. Ferrero, R. Valencia, B. Vanthournout, Final report on embedded software estimation and model generation, Tech. Rep. COMPLEX/PoliMi/R/D2.2.2/1.0, COMPLEX project deliverable (Jan. 2012).
URL `http://complex.offis.de/docs/31`

[69] C. Brandolese, W. Fornaciari, Software Energy Optimization Through Fine-Grained Function-Level Voltage and Frequency Scaling, in: International Conference on Hardware/Software Codesign and System Synthesis, CODES+ISSS'2012, Tampere, FI, 2012.

[70] D. Lorenz, P. A. Hartmann, K. Grüttner, W. Nebel, Non–invasive Power Simulation at System–Level with SystemC, in: International Workshop on Power and Timing Modeling, Optimization and Simulation, PATMOS'2012, Newcastle upon Tyne, UK, 2012.

[71] D. Lorenz, K. Grüttner, N. Bombieri, V. Guarnieri, S. Bocchio, From RTL IP to Functional System-Level Models with Extra-Functional Properties, in: International Conference on Hardware/Software Codesign and System Synthesis, CODES+ISSS'2012, 2012.

[72] S. Bocchio, P. A. Hartmann, D. Lorenz, D. Quaglia, Final report and tools on platform IP components estimation and model generation, Tech. Rep. COMPLEX/ST-I/P/D2.3.2/1.1, COMPLEX project deliverable (May 2012).
URL `http://complex.offis.de/docs/34`

[73] SystemC Network Simulation Library v.2 (2012).
URL `http://sourceforge.net/projects/scnsl`

[74] P. Sayyah, M. Lazarescu, D. Quaglia, E. Ebeid, S. Bocchio, A. Rosti, Network-aware Design-Space Exploration of a Power-Efficient Embedded Application, in: International Conference on Hardware/Software Codesign and System Synthesis, CODES+ISSS'2012, Tampere, FI, 2012.

[75] P. Sayyah, F. Stefanni, L. Lavagno, D. Quaglia, SystemC model generation for realistic simulation of networked embedded systems, in: 15th Euromicro Conference on Digital System Design, DSD'2012, Cesme-Izmir, TR, 2012.

Chantal Ykman-Couvreur is mathematician. She first worked at PHILIPS Research Laboratory of Belgium, from November 1979 until June 1991. Her main activities were concentrated on information theory and coding, cryptography, and multi-level logic synthesis for VLSI circuits. She joined IMEC in September 1991. At IMEC, her main research projects have been synthesis of asynchronous control circuits, dynamic memory management and task concurrency management at the system level, multi-processor SoC design. She is currently active in run-time management for embedded multi-core platforms.

Davide Quaglia received his PhD in Computer Engineering from Politecnico di Torino (Italy) in 2003. Currently he is Assistant Professor at the Computer Science Department of the University of Verona (Italy). His current research interests include Networked Embedded Systems, Networked Control Systems, Cyber-Physical Systems. He is also co-founder and active project leader of EDALab s.r.l., a spin-off company of the University of Verona.

Prof. Eugenio Villar got his Ph.D. in Electronics from the University of Cantabria in 1984. Since 1992 is Full Professor at the Electronics Technology, Automatics and Systems Engineering Department of the University of Cantabria where he is currently the responsible for the area of HW/SW Embedded Systems Design at the Microelectronics Engineering Group. His research activity has been always related with system specification and modeling. His current research interests cover system specification and design, MPSoC modeling and performance estimation using SystemC and UML/Marte. He is author of more than 130 papers in international conferences, journals and books in the area of specification and design of electronic systems. Prof. Villar served in several technical committees of international conferences like the VHDL Forum, Euro-VHDL, EuroDAC, DATE, VLSI-SoC and FDL. He has participated in several international projects in electronic system design under the FP5, FP6 and FP7, Itea, Medea-Catrene and Artemis programs. He is the representative of the University of Cantabria in the ArtemisIA JU.

Francisco Ferrero studied Telecommunications Engineering at the Carlos III University of Madrid. He has been involved in several projects related with avionics and embedded software for the aeronautics industry. He is the technical responsible in IDEFIX, an European R&D programme for the definition, analysis and exercise of the development process for an Integrated Modular Avionics (IMA) system based on STANAG 4626 standards. Currently he is involved in various activities related to the development, integration and qualification of ASAAC applications, and analysis of tool frameworks to support IMA development. He is experienced with modelling tools for analysing of system configuration and resource provision.

Kim Grüttner received a Diploma degree in Computer Science from Carl von Ossietzky University Oldenburg, Germany in 2005, and is currently working on his PhD thesis on "Application Mapping and Communication Synthesis for Object-Oriented Platform Based Design". He joined OFFIS research institute in 2005 and worked in the European projects ICODES and ANDRES. Since 10/2008 he is manager of the "Hardware/Software Design Methodology" group, and the technical coordinator of the COMPLEX European integrated project. His research interests are system-level design languages, system synthesis methodologies, and modeling of extra-functional properties at system-level.

William Fornaciari is Associate Professor at Politecnico di Milano, Dipartimento di Elettronica e Informazione. He published six books and over 150 papers in international journals and conference proceedings, collecting four best paper awards, one certification of appreciation from IEEE and holds two international patents on low power design solutions. Since 1993 he is member of program and scientific committees and chair of international conferences in the field of computer architectures, EDA and system-level design.
Since 1997 has been involved in 11 EU-funded international projects and he has been part of the pool of experts of the Call For Tender No.964-2005 – WING – Watching IST INnovation and knowledge, studying the impact of FP5 and FP6 expenditure for the EC, in the perspective to support the identification of FP7 and Horizon2020 research directions.
He is also project reviewer for the European Commission and national research bodies in Europe. During the last 20 years he has worked as consultant for both management and technical issues for many ICT industries, gaining a relevant experience in technology transfer and product development.
His current research interests include embedded systems design methodologies, real-time operating systems, energy-aware design of sw and hw, runtime management of resources, reconfigurable computing and wireless sensor networks, design and optimization of multi-core systems, NoC design and optimization, reliability.

Wolfgang Nebel holds a Dipl.-Ing. degree in Electrical Engineering from the University of Hanover and a Dr.-Ing. degree from the Computer Science Department of the University of Kaiserslautern. In 1987 Nebel joined Philips Semiconductors, Hamburg, and worked as software engineer, CAD project manager and finally became CAD software development manager. In 1993 he was appointed to the professorship VLSI design at the department of computer science at the Carl von Ossietzky University of Oldenburg. From 1996 to 1998 he served as Dean of his department. Additionally since 1998 Nebel has been a member of the executive board of the OFFIS research center, an institute for information technology which is associated with Oldenburg University. From January 2001 December 2002 Nebel served as vice-president of Oldenburg University. Since June 2005 he has been chairman of the OFFIS - Institute for Information Technology. Wolfgang Nebel is and has been involved in several international conferences as program chair or a general chair. He is also active in several additional program committees and professional organizations (IEEE Fellow). His research interest are methodologies and tools for embedded system design, in particular: object oriented HW/SW specification and synthesis as well as design for low power.