Power-proportional modelling fidelity

A. Rafiev, F. Xia, A. Iliasov, R. Gensh,
A. Aalsaud, A. Romanovsky, A. Yakovlev
– Newcastle University, UK
{ashur.rafiev, fei.xia, alexei.iliasov, rem.gensh,
a.m.m.aalsaud, alexander.romanovsky, alex.yakovlev}@ncl.ac.uk

Abstract

Traditional hierarchical modelling methods tend to have layers of abstraction corresponding to naturally existing layers of concern in multi-level systems. Although convenient, this is not always optimal for analysis and design. For instance, parts of a system which are in the same layer may not contribute to the same degree on some metric, e.g. system power consumption. To moderate the modelling, analysis and design effort, and potentially runtime control overhead for models used at runtime, less significant parts of the system should be studied at higher levels of abstraction and more significant ones with more detail. Concentrating on system power consumption, this paper presents Order Graphs (OGs), which have a clear hierarchical structure, but provide straightforward vertical zooming across multiple layers (orders) of model fidelity, resulting in the discovery of power-proportional cuts that run through different orders to be analysed together in a flat manner. Stochastic Activity Networks (SANs), a good flat modelling method, is suggested as an example of studying technique for cuts discovered with OGs. A series of experiments on an Odroid development system consisting of an ARM big.LITTLE multi-core structure provides initial validation for the approach.

1 Introduction

Systems with large scale concurrency and complexity, e.g. computation systems built upon architectures with multiple and increasingly many processing cores with heterogeneity among the components, are becoming more popular and common-place [3]. The hardware motivations are clear, as concurrency scaling can help delay the potential saturation of Moore's Law with current and future CMOS technology and better use the opportunities provided by the technology scaling. In this environment, software designs are increasingly focused towards greater concurrency and mapping to such many-core hardware [9].

Both hardware and software of these types tend to form hierarchical structures, for instance, the levels of detail in hardware include the entire spectrum from transistors to gates to function blocks to entire CPUs to multiple CPUs

with supporting logic, memory, etc. For system designers, software (e.g. applications), operating systems, and the platforms on which these are run also form natural design layers with clear boundaries between the layers. Such structures are usually conveniently modelled with traditional hierarchical modelling methods, with the modelling levels of abstraction corresponding to these system layers of concern [7].

This is, however, not always optimal for analysis, design and runtime management. In most cases these require the modelling of particular parameters and the "modelling fidelity" should, ideally, be determined by the parameter(s) under study [12]. For instance, when a part of a system makes a crucial contribution to the power consumption of the entire system and small changes may have a significant effect, it pays to study it in detail, i.e. at some lower layer of abstraction. On the other hand, to moderate the modelling, analysis and design effort, and potentially runtime overhead for models that need to be used in runtime, other less significant parts of the system should be studied at higher levels of abstraction. When this "centre of gravity" of system operation concerning power can dynamically move around the system, traditional hierarchical modelling methods are ill positioned for efficient representation.

Hierarchical methods, because of their complexity, are usually less straightforward to use than flat representations. Petri nets [2], which exemplify flat modelling methods, have extremely simple semantics and offer conveniences in reasoning, proofing and other aspects of analysis, a quality shared by other flat modelling methods. But when the modelling needs span multiple layers in a hierarchy it becomes somewhat difficult to adopt flat methods as study tools.

In this paper, we present Order Graphs (OGs), a model that has a clear hierarchical structure, but provides straightforward vertical zooming across multiple layers (orders) of model fidelity independently in different regions of a model, resulting in the isolation of cuts that run through different orders. Such a cut can then be analysed in a flat manner using existing flat modelling methods. These cuts can easily move with the system's operation based on the power consumption of their parts and thus always concentrate the most appropriate amount of modelling effort on each part of the system resulting in the optimal modelling fidelity for each part at each state. The ideal scenario will lead to true power-proportional modelling fidelity and thus effort of analysis.

In this paper, we adopt Stochastic Activity Networks (SANs) [11], a quantitative derivative of Petri nets with both probabilistic and deterministic representation facilities, as an example flat technique for quantitative studies. A series of experiments on an Odroid development system consisting of an ARM big.LITTLE multi-core structure [6] provides initial validation for the approach.

Section 2 describes our modelling approach including OGs and their use in the power-proportional modelling fidelity approach. Section 3 describes the application example Odroid system with platform modelling in OGs and component modelling in SANs and a series of experiments which supports the approach. Section 5 includes discussions.

2 Modelling Approach

2.1 Resource-driven Modelling

The central subject of our method is the study of a computational platform comprising a number of diverse resources and the way resources may be handled in order to realise a computation. A resource is in this case an indivisible element required by the system in order to change its state, and it is defined by its function and availability in relation to this transition. With the word "resources" we make the point that we do not exclude computation, communication, or other facilities, e.g. energy and time. A resource graph is a relation graph, where each vertex represents a single resource and each edge represents a relation or dependency between two resources [10].

In many real-life systems the dependencies between the resources do not have to be maintained all the time in order for the system to function normally. In fact, for some systems the functionality requires switching dependencies on and off. In this paper, however, we focus on a static resource view of the system, showing all possible resources and dependencies. This way of modelling is focused on exploring the structure of the system and does not provide the means to estimate quantitative properties of the system. For quantitative analysis we can refer use other methods, for example, SANs [11].

2.2 Modelling Hierarchy in Order Graphs

An underlying approach for having adjustable fidelity in the models relies on different levels of abstraction. Naturally, these layers have to be consistent with each other. Modelling across layers of abstraction is usually called hierarchical modelling.

Figure 1(a) shows the conventional approach to model hierarchies, which uses tree structures [7]. Each node of the higher layer zooms into a subgraph in the lower layer. Consequently, an edge between two nodes becomes multiple edges between the corresponding subgraphs. The notation used in the diagram is based on Zoom Structures [5]. A convenient way to display graph hierarchies is zoom views, combining verticality and horizontality: each abstraction layer is a horizontal view of the system; the information on how different layers interlink is represented using vertical directed arcs.

By the definition of resource graphs, anything can be considered a resource. Can we say that the edges of a graph are also resources? It is actually true, and this contradiction is explained and solved by Order Graphs (OGs). As an example, let's imagine that Figure 1(a) models a network interaction, where a is a server and b is a client. On the very abstract level we do not care about the structure of the network at this level of abstraction, we just need to know that the client and the server are connected somehow, thus we model this entire system as the client and the server connected directly with a single dependency. However, in a detailed model we can no longer ignore the network protocols and have to introduce it at least as a single resource node as shown in Figure 1(b).

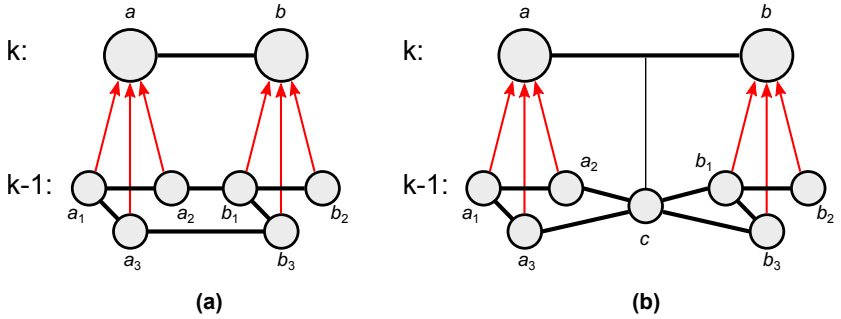

Figure 1: Conventional hierarchy representation (a) compared to Order Graphs (b); k is the higher level of abstraction and $k-1$ is the lower level.

A distinct property of the proposed Order Graph modelling approach is that a high-order edge is at the same time a node at the lower order. In this case we say that the node *supports* an edge, while in fact this is the same entity viewed from the different abstraction levels. In real-life systems, any dependency is always supported by a resource of some kind, and this "fractal" structure goes down to the smallest details, including atoms and below. Of course, we do not want to include all these in the model, so we had to flex the rule by saying that an edge is either supported by a resource at the lower layer or stays an edge (brought forward).

Each layer in a hierarchy provides a different modelling fidelity when mapped into a quantitative model, however some parts of the system have less importance than others, for instance due to a smaller contribution to the studied metric. In the proposed approach we analyse the system using cross-layer *cuts*. A cut is defined as a subgraph of OG that includes elements from different layers but only if they are not vertically related, so the cut itself is a flat graph. For example, two possible cuts in Figure 1(b) are (a_1, a_2, a_3, c, b) and (a, c, b_1, b_2, b_3).

3 Case Study

In this section, we use an example system and its modelling to lead towards our power-proportional fidelity modelling flow.

3.1 Platform Description

The Odroid XU3 board [1] is a small Octa-Core computing device implemented on energy-efficient hardware. The board can run different versions of OS, for example Ubuntu 14.04 or Android 4.4.

The main component of Odroid XU3 is the 28nm Application Processor Exynos 5422. This System-on-Chip is based on the ARM big.LITTLE architecture [6] and consists of a high performance Cortext-A15 quad core processor block, a low power Cortex-A7 quad core block, a Mali-T628 GPU and

2GB DRAM LPDDR3. The board contains four real time current sensors allowing the measurement of power consumption on the four separate power domains: big CPU, little CPU, GPU and DRAM. There are also four temperature sensors for the A15 processors and one for the GPU.

On the Odroid, for each power domain, the supply voltage (Vdd) and clock frequency can be tuned through a number of pre-set pairs of values, allowing dynamic frequency scaling (DFS) when the frequency is between 200MHz and 800MHz (the Vdd stays constant in this region) and dynamic voltage and frequency scaling (DVFS) [4, 8] when the frequency is 800MHz and above.

3.2 Platform Model in Order Graphs

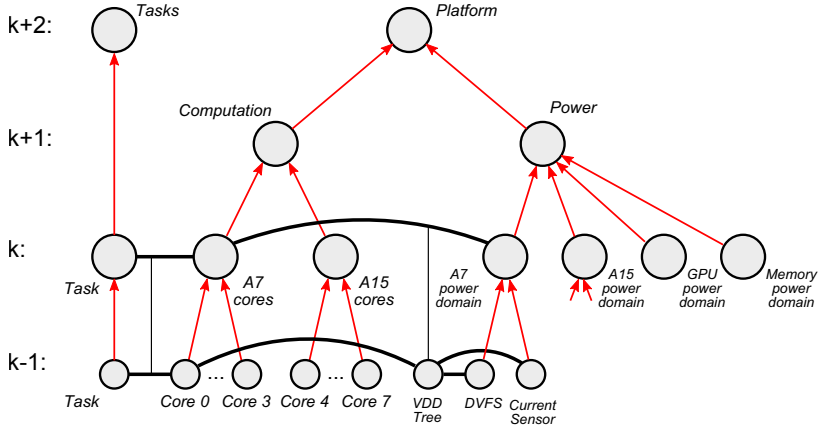

Figure 2: Order Graph model of running tasks on Odriod platform (some horizontal dependencies are omitted).

In this case study we focus on modelling power consumption of the platform. Two major contributors are task affinities (which task runs on which core) and DVFS. Figure 2 shows the OG model of the system. At the higher levels of abstraction, the system is represented as a set of tasks running on a platform, which in turn consists of a computation component and a power component. The computation resource is provided by A7 and A15 cores, which appear in the lower orders, and the power resource is divided into four power domains, as described in Section 3.1. For clarity, some of the horizontal edges on this diagram are hidden: every core is actually connected to the corresponding Vdd tree and to the task node, etc.

3.3 Platform Model Components

SANs are an extension to Generalized Stochastic Petri Nets (GSPNs) and a more expressive representation language. The SANs formalism provides a general

way of specifying the enabling of an activity or transition, a general way of specifying a completion (firing) rule, a method of representing zero-time events (hence including deterministic as well as stochastic behaviours), a method of representing probabilistic choice in addition to probabilistic delay provided by GSPNs. It also provides state-dependent parametric values and general delay distributions on activities.

With the Odroid platform, the major controls available for runtime power management are the DFS and DVFS of the core blocks (power domains), the activation and inactivation of individual cores, and the mapping of specific threads or tasks to individual cores. Figure 3 shows several possible ways of modelling these choice-based decisions. The task scheduling models describe an environment where tasks are organized into three queues, one going to the A15 processors, one going to the A7 processors, and one with non-deterministic designations. The algorithm in these models sets tasks to either the A15 or the A7 queue. This is done by specifying the logic for the output gates to decrement and increment the task queue markings accordingly when a transition fires. For instance, if transition A15 fires, one of the tasks in one of the other queues is moved to the A15 queue. The two models have different levels of fidelity in their representation; Figure 3(b) is a more deterministic case; Figure 3(a) is a more probabilistic case. The modelling and analysis costs/efforts of these models are related to their representational fidelity. The DVFS model in Figure 3(c) is the most probabilistic, has the least fidelity, and is the easiest to use, but higher fidelity versions up to fully deterministic can also be constructed. In these models, the task scheduling and DVFS transitions are assumed to be triggered by other sub-nets representing the controllers, which are not included here.

The other crucial issue to be modelled for this system, when we talk about system power consumption, is processing, i.e. the execution of threads/tasks in the cores. The fundamental processing element model is shown in Figure 4(a). Here the place Capacity represents the unused capacity of a processing element (e.g. a core), and the place Processing represents the current number of tasks being executed in the core. If it is a single core, the sum of markings of these two places represents the pipeline depth or multi-threading capability of the core. If there are multiple cores in this model, the sum of markings represents the entire block's multi-threading capability.

Different levels of fidelity are possible with this representation. For instance the degree of probabilistic vs. deterministic can be tuned for a more or less fuzzy representation. We may decide to model part of a core (i.e. a multiplier), an entire single core, a core-block, or the entire Odroid chip with one of these sub-nets. When setting up a more detailed model with higher fidelity, we may need to distinguish how a processing element behaves with different types of tasks (see Figure 4(b)), as shown in subsequent sections the Odroid cores consume different amounts of power when dealing with different tasks. With more fuzzy representations, such issues may be covered by probabilities.

Once a cut has been determined using the OG model, a flat SANs model covering the entire system can be made with different levels of fidelity for different parts. This will be a flat model with power-proportional fidelity and effort.

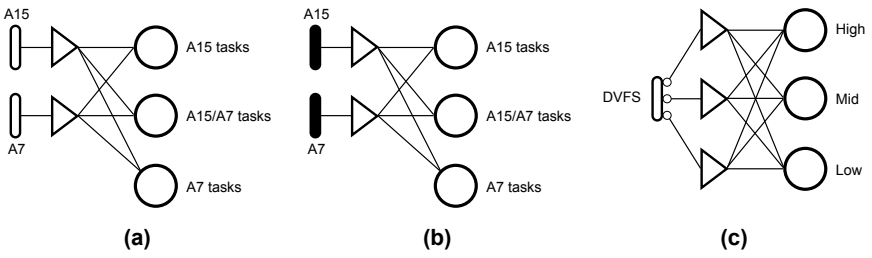

Figure 3: SANs models for stochastic (a) and deterministic (b) affinity, and DVFS (c).

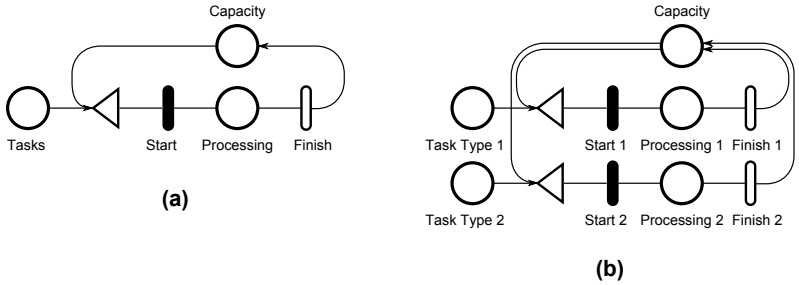

Figure 4: SANs models for task execution

3.4 Model Characterisation Experiments

Experiments with the Odroid platform were carried out in order to understand the power consumption under different operation frequencies and voltages. The low-power A7 quad core block can scale its frequencies from 200MHz to 1400MHz, whilst the performance-oriented Cortex A15 block has a range of frequenciy from 200MHz to 2000MHZ. The frequency of each block can be changed independently using OS commands and the system scales the operating voltage of the block to fit the chosen frequency. The on-chip sensors allow voltage, current and power for each processor to be measured in real time.

In our characterization experiments, firstly the above parameters were measured without any additional workload, with only the OS running. Then the same parameters were measured for each core with application threads running. We experimented with the typical Linux stress task, i.e. running square root calculations repeatedly, and in addition, other computations including logarithm calculations and the four arithmetic operations.

Another important experiment is the measurement of the same parameters with some of the cores in each block disabled: Odroid allows from one to four of the A15 being disabled and from one to three of the A7 to be disabled. At least one A7 must be running for the OS to be alive.

In these experiments, it was observed that an A15 consumes four times or more power than an A7 when both are running at the same frequency, up to

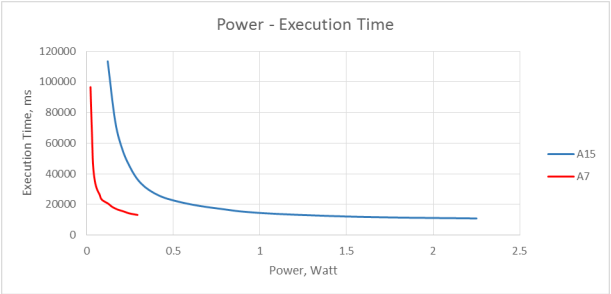

Figure 5: Measured power to execution time

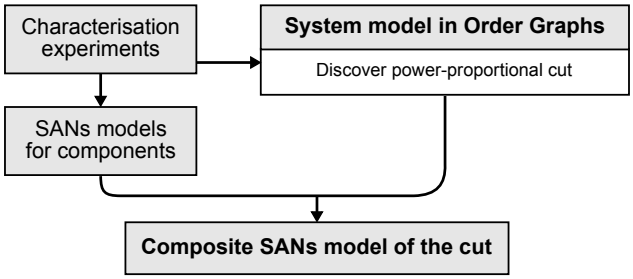

Figure 6: Modelling workflow

an order of magnitude more power when both are running at the same voltage. Figure 5 shows the relationship between power consumption and the execution time for the two types of cores on the average running a range of different types of tasks.

These radically different performance and power figures, and their complex relations to the different tasks being executed in a core, validate the approach promoted in this paper. For instance, when certain tasks are mapped to the A7 block, because of the relatively light power demand of these cores we may be able to afford to model such processing with less fidelity, i.e. using a more probabilistic model and/or using a more structurally fuzzy model. For instance, when the A15 block is also running, it may be a good idea to not represent individual A7 cores but to cover the entire A7 block with a single model of the type in Figure 4(a).

4 Modelling flow

We propose a modelling flow based on the exploration of power-proportional cuts using OGs. This modelling flow will result in both fidelity and effort to be as proportional to power as possible (or, instead of power, any metric or combination of metrics). The flow is shown in Figure 6. Characterisation experiments on the components of the system provide information on the crucial parameters

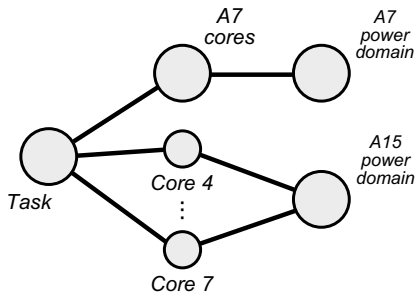

Figure 7: Proposed cross-layer cut for power-proportional modelling.

and establish the elementary model for each component. The parameters would help to identify appropriate cuts in the OG model for further analysis. The flat model used for this analysis is obtained from composing separate elementary models according to the OG cut.

The method of using hierarchy cuts is non-destructive, meaning that the model can be easily re-arranged and adapted for a different fidelity distribution. This is especially helpful if the model is used in run-time management of the systems working in various "modes". Hence the proposed approach appears advantageous compared to a single flat model.

In Section 3 we use SANs just as an example of possible quantitative modelling of the system's metrics. Hence we do not focus on evaluating the absolute values. The goal of the study is to demonstrate the flexibility of the method w.r.t. a heterogeneous platform and a variety of tasks.

Using this modelling flow and based on experimental data from the Odroid, for a certain modelling fidelity we may need to represent each A15 core with a model of the type in Figure 4(b), with two types of tasks - CPU heavy and memory heavy, and five levels of DVFS resolution. For the same level of fidelity we can represent the entire A7 block with a single sub-net of the type in Figure 4(a) without task and DVFS differentiation. The corresponding OG cut, shown in Figure 7, represents a model size savings of well over 40% without any impact on the practical modelling fidelity so long as power consumption is concerned.

5 Conclusions

This paper proposes a flexible fidelity modelling approach to complex systems and uses power as an example metric to illustrate the concept. The result is a modelling flow with power-proportional fidelity. This method is centred on OGs, a new formalism with facilities for independent vertical zooming among different parts of a model, and the straightforward exploration and discovery of appropriate power-proportional cuts. These cuts are then very suitable for exploration, reasoning and analysis with established qualitative and quantitative

flat representation methods. Here SANs are used as an example for this type of exploration. Experiments with a heterogeneous system with multiple cores and power domains as well as different types of computation tasks help validate and further motivate the approach.

The future work would focus on applying the proposed method to the development of an intelligent run-time power control for heterogeneous many-core systems.

Acknowledgement This work is supported by EPSRC grant EP/K034448/1.

References

[1] Odroid XU3. http://www.hardkernel.com/main/products.

[2] G. Balbo. *Formal Methods for Performance Evaluation*, volume LNCS4486, chapter Introduction to Generalized Stochastic Petri Nets, pages 83–131. Springer, 2007.

[3] S. Borkar. Thousand core chips: A technology perspective. In *Proceedings of the 44th Annual Design Automation Conference*, DAC '07, pages 746–749, New York, NY, USA, 2007. ACM.

[4] A. P. Chandrakasan, S. Sheng, and R. W. Brodersen. Low power CMOS digital design. *IEEE Journal of Solid State Circuits*, 27:473–484, 1995.

[5] A. Ehrenfeucht and G. Rozenberg. Zoom structures and reaction systems yield exploration systems. In *IJFCS*, pages 275–306, 2014.

[6] P. Greenhalgh. *big.LITTLE Processing with ARM Cortex-A15 & Cortex-A7 – Improving Energy Efficiency in High-Performance Mobile Platforms.* ARM, 2011. White Paper.

[7] B. Kumar and E. S. Davidson. Computer system design using a hierarchical approach to performance evaluation. *Commun. ACM*, 23(9):511–521, September 1980.

[8] E. Le Sueur and G. Heiser. Dynamic voltage and frequency scaling: The laws of diminishing returns. In *Proc. of 2010 Intern. Conf. on Power Aware Computing and Systems*, HotPower'10, USA, 2010.

[9] Yves Lhuillier et al. HARS: A hardware-assisted runtime software for embedded many-core architectures. *ACM Trans. Embed. Comput. Syst.*, 13(3s):102:1–102:25, March 2014.

[10] A. Rafiev et al. Studying the interplay of concurrency, performance, energy and reliability with ArchOn – an architecture-open resource-driven cross-layer modelling framework. In *Proc. to ACSD*, 2014.

[11] W.H. Sanders and J.F. Meyer. *Lectures on Formal Methods and Performance Analysis*, volume LNCS2090, chapter Introduction to Generalized Stochastic Petri Nets, pages 315–343. Springer, 2001.

[12] Bo Wang et al. End-to-end power estimation for heterogeneous cellular LTE SoCs in early design phases. In *Power and Timing Modeling, Optimization and Simulation (PATMOS), 2014 24th International Workshop on*, pages 1–8, Sept 2014.