Microelectronics System Design Research Group
School of Electrical and Electronic Engineering
Merz Court
Newcastle University
Newcastle upon Tyne, NE1 7RU, UK

http://async.org.uk/

Many-core Architectures: a Study in Tiles and Colour

Andrey Mokhov

andrey.mokhov@ncl.ac.uk

NCL-EEE-MSD-MEMO-2013-006

April 2013

Abstract

This memo discusses a simple many-core architecture for general-purpose computing, which is capable of the run-time adaptation towards desired corners of the performance-energy-reliability tradeoff space. In this architecture, processing cores are identical in their functional characteristics, but internally they are implemented differently in a fixed number of design styles, each optimised for a particular non-functional criterion, e.g., performance, energy-efficiency, reliability, etc.

Similar to an RGB-monitor capable of reproducing a rich variety of colours using elements of only three basic colours (red, green and blue), such a many-core system can gracefully adapt to a broad range of operating conditions by changing the proportion of activated cores of each type.

1 Introduction

In the recent years we have witnessed a steady trend towards increasing the number of processing cores across all segments of computing from high-performance to low-power embedded systems. The key reason for that is higher energy-efficiency that can be achieved by multiple slower cores running in parallel compared to a single blazingly fast core. It is envisaged that future computing systems will comprise hundreds of cores and beyond. We will call such systems 'many-core' to differentiate them from 'multi-core' systems having only several processing cores (still more common today).

The many-core computing paradigm presents a number of challenges with respect to hardware design, software engineering, and run-time core management to provide non-functional guarantees for performance, energy-efficiency and reliability. Since exploration of all possible many-core architectures is infeasible, we propose to start with an intentionally simple architecture capable of demonstrating most of the many-core related phenomena; if necessary, the architecture can later be further elaborated, e.g., by allowing the functional diversity.

2 Architecture

In this memo we propose a simple many-core architecture, which is capable of the run-time adaptation towards desired corners of the performance-energy-reliability tradeoff space. In this architecture, processing cores are identical in their functional characteristics, but internally they are implemented differently in a fixed number of design styles, each optimised for a particular non-functional criterion, e.g., performance, accuracy, energy-efficiency, reliability, etc. One might consider giving a fancy name to this architecture, for example, *Functionally Homogeneous Structurally Heterogeneous (FHSH)*; however, throughout this memo we will refer to it plainly as 'the architecture'.

2.1 Rationale

The architecture is intentionally simple. Each core is indistinguishable from others in terms of functionality, therefore the operating system (OS) has no 'compatibility' constraints for different task-core pairs, which significantly simplifies task scheduling. It is also quite pragmatic from the implementation point of view: each core is synthesised with optimisations chosen according to the core's type (which can yield a significant improvement with respect to the selected criterion, e.g., speed); moreover, the cores can be arranged in a regular 2D or 3D silicon mesh, simplifying layout and interconnect (see Section 3 for discussion of layout aspects).

For simplicity, we do not consider dynamic voltage and frequency scaling (DVFS) [2] in this setting. Instead, each core type may operate only on a particular combination of supply voltage and frequency. If one assumes that *the number of cores is significantly higher than the number of core types,* which is likely to hold in future, then it is possible to achieve an equivalent average voltage and frequency at the macro scale by activating cores in an appropriate proportion. This does not only simplify reasoning on the architecture, but also allows to remove overheads associated with DVFS. See [3] for an in-depth comparison of DVFS and dynamic activation of different fixed-parameter cores.

Despite simplicity, the architecture allows fine-grain modelling and control over the following non-functional characteristics of the overall computing system:

1. *Performance:* by scheduling tasks on fast or slow cores, or a combination thereof, the OS can achieve an optimum Quality-of-Service (QoS) depending on the current application and operating conditions.

2. *Energy-efficiency:* if energy-per-task is more important than performance, the OS can prefer more energy-efficient type of cores, deactivating the high-performance ones.

3. *Reliability:* since the cores have identical functionality it is possible to re-run a task which has failed on a particular core on another, more reliable one. Due to the available diversity of non-functional characteristics, the OS has a rich choice of possible scheduling strategies with respect to reliability.

4. *Power proportionality:* a computation system, for it to be considered power-proportional [4], has to keep power consumption and computation load proportional to each other. That is, an idle system would ideally consume no power, and vice versa, given a low power budget a system would adapt itself by reducing its computation flow and thereby lowering the delivered Quality-of-Service (QoS), yet still remaining functional [6]. This can be achieved in this architecture by keeping the number of active cores proportional to the current load.

5. *Survival:* a small number of cores may be designed to withstand particularly harsh operating conditions, e.g., very low and/or unstable voltage supply. These cores may be very inefficient due to thorough error correction and redundancy; their only purpose is to be activated in situations when no other type of cores can survive.

6. *Dark silicon and ageing:* the OS may be aware that to balance heat dissipation and/or wear of individual cores, it is beneficial to alternate their activation in certain cyclic or random patterns over time (see Section 3).

2.2 Parameters

The architecture has the following parameters:

- N is the number of processing cores.
- T is the number of different implementation styles, or simply *types* of the cores.
- c_k is the number of cores of type k ($1 \leqslant k \leqslant T$). We assume that $\forall k, c_k \geqslant 1$ and $\sum_k c_k = N$.

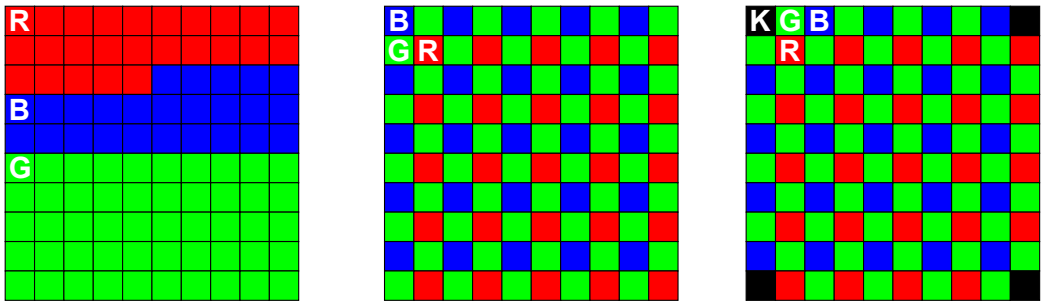
For example, ARM's big.LITTLE [1] can be considered the simplest non-trivial instance of this architecture with parameters $N = 2$, $T = 2$, $c = (1,1)$, while NVIDIA's Tegra 4 chip [5] is a more sophisticated instance with parameters $N = 5$, $T = 2$, $c = (1,4)$. See [3] for an example with $T = 3$ types of cores and $N = 3$, $c = (1,1,1)$.

3 Chip geometry: from cores and types to tiles and colour

Cores of different types are likely to require different supply voltages; moreover, there will be different power and temperature effects associated with their activation. Therefore, it is necessary to consider different possible chip layouts.

3.1 Power supply and dark silicon

Consider a many-core system with the following parameters: $N = 100$, $T = 3$, $c = (25, 50, 25)$. We will use colours to represent cores of different types: red will stand for 25 cores of type 1 (high performance, 1.0V power supply), green will stand for 50 cores of type 2 (medium performance, 0.85V power supply), and blue will stand for 25 cores of type 3 (low power, 0.7V power supply). Two possible layouts are shown in Figure 1.



(a) Trivial layout (b) Bayer layout (c) Adding cores for survival

Figure 1: Many-core chip layouts (letters denote colours for black and white version)

Clearly, layout in Figure 1(a) is less preferable because of non-balanced energy consumption and heat distribution: the upper part of the chip will get overheated if all cores of type 1 are activated. In fact, the modern semiconductor technology is capable of putting so many transistors in a tiny silicon area that it is not possible to switch all of them on at the same time – parts of the chip must stay off to avoid overheating (so called *dark silicon*). Therefore, one might try to arrange core types in a manner similar to the famous Bayer colour filter used in photography, see Figure 1(b). Surprisingly, it fits the unusual purpose very well: high power components are separated from each other by medium and low power ones; in addition, regularity of the layout simplifies design the power supply network.

3.2 Survival

Figure 1(c) shows layout of a many-core system with parameters $N = 100$, $T = 4$, $c = (24, 48, 24, 4)$. In addition to the core types listed in the previous section, 4 high-resilience cores of black colour are added – they will be capable of operating in the survival mode under very low and/or unstable voltage supply. Locating these additional cores at the edge of the chip allows to simplify routing for a dedicated power supply. Another possible layout is to put the black tiles together in one of the corners in a 2x2 shape. Shaping and layering of tiles of different colours may allow achieving an optimal layout with respect to costs of communication between 'master' and 'slave' cores.

4 Analytical characterisation

The presented approach to modelling many-core systems allows to apply Markov analysis for analytical characterisation of various system properties, such as reliability, performance, survival, etc.

Let a_k denote the number of currently active cores of type k ($1 \leqslant k \leqslant T$). Since $0 \leqslant a_k \leqslant c_k$ the following number of combinations C are possible:

$$C = \prod_{1 \leqslant k \leqslant T} (c_k + 1).$$

Note that in reality a_k may be limited not by the number of available cores c_k but by heat dissipation limits. In the age of dark silicon a_k could be significantly smaller than c_k; this will reduce the number of possible combinations.

We can associate a system state with each such combination. If we further assume that transitions between the states occur with probabilities independent of the previous history then the resulting system can be modelled as a Markov process, and analytical characterisations of some of its parameters can be readily derived.

For example, let there be three types of cores (red, green, and blue as in Section 3.1) and let the cores have probabilities of failures p_1, p_2 and p_3 (the probabilities are different due to different supply voltages). In this case, a system can transition from state (a_1, a_2, a_3) to state $(a_1 - 1, a_2, a_3)$ with probability p_1 (assuming $a_1 > 0$). If the repair rate of cores of type 1 is equal to r_1 then the opposite transition will occur with probability $1/r_1$. This allows to analytically derive such characteristics as average core utilisation (proportion of time spent on computation vs time spent on repair), MTBF (if global system failure corresponds to a situation when a critical number of cores fails simultaneously) and others.

5 Conclusions

A simple many-core architecture is proposed as an experimental platform. We consider it to be useful for exploring performance-energy-reliability tradeoffs in modern and future many-core systems.

The platform is discussed from a purely structural angle. Behavioural and control aspects are to be studied in future over the duration of EPSRC PowerProp (Hardware/Software Mechanisms for Cross-Layer Power Proportionality) and PRiME (Power-efficient, Reliable, Many-core Embedded systems) projects.

References

[1] ARM. Big.LITTLE processing with ARM Cortex-A15 & Cortex-A7, 2011.

[2] Stephan Henzler. *Power Management of Digital Circuits in Deep Sub-Micron CMOS Technologies (Springer Series in Advanced Microelectronics)*. Springer-Verlag New York, Inc., 2006.

[3] T. Ishihara. Real-time power management for a multi-performance processor. In *SoC Design Conference (ISOCC), 2009 International*, pages 147–152, 2009.

[4] David Meisner, Christopher M. Sadler, Luiz André Barroso, Wolf-Dietrich Weber, and Thomas F. Wenisch. Power management of online data-intensive services. In *Proceedings of the 38th annual International Symposium on Computer Architecture (ISCA'2011)*, pages 319–330, 2011.

[5] NVIDIA. Variable SMP – A Multi-Core CPU Architecture for Low Power and High Performance, 2011.

[6] Alex Yakovlev. Energy-modulated computing. In *Design Automation and Test in Europe (DATE) conference*, pages 1340–1345, 2011.