Different approaches for power efficient heterogeneous multi-core embedded systems: an overview

Dennis Benders
Student Embedded Systems
Delft University of Technology
Email: D.Benders@student.tudelft.nl
Student number: 4451546

Bastiaan Burgers
Student Embedded Systems
Delft University of Technology
Email: D.B.J.Burgers@student.tudelft.nl
Student number: 4455347

Ivo van Straalen
Student Embedded Systems
Delft University of Technology
Email: I.vanStraalen@student.tudelft.nl
Student number: 4467531

***Abstract*—Nowadays, a lot of (wearable) devices contain multi-core embedded systems. The biggest problem they are facing is to increase performance and satisfy real-time constraints, while still maintaining their relatively long battery life. Therefore, a lot of research is going on in order to reduce the power consumption of these devices. For this paper, 4 different solutions have been chosen to elaborate on. The solutions cover different aspects of the processor architecture to be optimised in terms of power consumption. From these solutions the reconfigurable cache architectures seem to show the highest increase in energy efficiency (upto 64% and 33% for the two parts of the solution, respectively). However, it is hard to compare the results of the different solutions, due to their different testing environments.**

I. Introduction

Heterogeneous multi-core processors are processors containing multiple domains with different properties to optimise certain processes. These kind of processors are often found in embedded systems, because they allow for a proper trade-off between power consumption and performance, which is highly desired for real-time embedded systems. [1]
The market for embedded systems in general is constructed in such a way that these devices are thrown away at the end of their lifetime. In many cases these devices did not even reach their end of life, which is quite strange. Especially in these days, sustainability plays an important role, so throwing devices away before the end of life should not be a common habit. However, most devices also require high-performance real-time constraints. This is to ensure smooth operations, but also to ensure that the device is not becoming outdated too fast.
Embedded systems should thus have a relatively good real-time performance and consume relatively little energy (or power; in this paper energy and power will mean the same) in order to ensure a long lifetime. In most cases, a trade-off has to be made between those two. It is easy to produce a power hungry high-performance device, but it becomes much more difficult when the power consumption should be substantially decreased, as motivated above.
Power consumption can be decreased in many different ways. The goal of this paper is to elaborate on and compare four different approaches to make heterogeneous multi-core processors more energy efficient, while still maintaining the required real-time performance level:

1) using different cache sizes to reduce miss rate and thus execution time;
2) using data prefetching;
3) by assigning different VF (Voltage Frequency) characteristics to the different cores, and
4) by using re-configurable cache architectures.

Important to note is that these solutions only provide a subset of all solutions available for power consumption decrease. The common element in these solutions is the design of the hardware architecture. For instance, task assignment algorithms also play a very important role in decreasing power consumption, but this is merely software related and not relevant in the context of this paper.
It can be concluded that the optimal solution could only be determined when dedicating architectures to certain applications. When speaking about multi-core embedded systems in general, it is hard to say which methods decrease power consumption the most. However, as will be shown later on, the reconfigurable cache architectures of solution 4 show the highest increase in energy efficiency.
The four different solutions will be described in Section II. Thereafter, these solutions are compared in Section III. Finally, Section IV will provide the conclusions derived in this paper.

II. Description of the Evaluated Solutions

This section describes four different solutions to optimise the energy efficiency in heterogeneous multi-core (embedded) systems, derived from the papers on which this overview is based. The hardware architecture design choices as well as the experimental results will be indicated per solution.

A. Solution 1: Reducing the overall cache miss rate using different cache sizes for heterogeneous multi-core processors

Often, Heterogeneous Multi-core Processors (HMPs) are investigated as a way to increase energy efficiency. In many cases, a technique called Dynamic Voltage Frequency Scaling

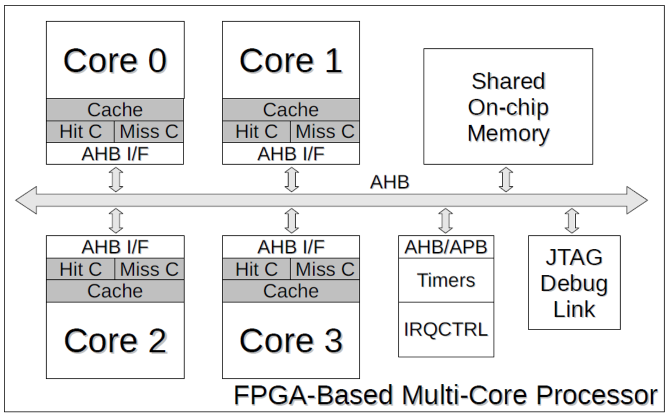

Fig. 1. Implemented multi-core system on a FPGA. Each core has two counters to measure the cache miss and hit rate. [2]

Fig. 2. Number of Cache Misses for each program and cache size, and the difference between steps. [2]

	Number of Cache Misses			
	pi	*jacobi*	*bubble sort*	*quick sort*
Cache 1KB	234	28,380,688	449,777,774	51,473,744
Cache 2KB	204	17,827,342	449,001,326	50,497,584
Cache 4KB	189	16,283,151	445,875,566	49,323,488
Cache 8KB	128	15,541,389	433,332,590	45,988,152
	Decreasing Step			
From 1KB to 2KB	**30**	10,553,346	776,448	976,160
From 2KB to 4KB	–	1,544,191	3,125,760	**1,174,096**
From 4KB to 8KB	–	**741,762**	12,542,976	–

Fig. 3. Summary of the different prefetchers. [3]

	cache hierarchy	prefetching degree	trigger L1	trigger L2
P1	L1 & L2	Dynamic	miss	Access
P2	L1	Static	miss	N/A
P3	L1 & L2	Dynamic	miss	Miss
P4	L2	Static	N/A	Miss
P5	L1 & L2	Dynamic	miss	Miss
P6	L2	Static	N/A	Access

(DVFS) is utilised for this purpose. However, de Abreu Silva et al. [2] study the effect of different cache sizes in such HMPs on the cache miss rate. Decreasing the cache miss rate will increase energy efficiency, as the execution time of the program will decrease, thus lowering energy consumption, while increasing performance. To investigate the benefits of different cache sizes, four applications were executed in different multi-core architectures, and the cache miss and hit rates were recorded. An FPGA was used to synthesise these HMPs. A schematic of the implemented multi-core system can be seen in Figure 1. Additionally, a prototype scheduler was developed. This will not be discussed in detail, however, because of the hardware focus.

The generated architecture contained four cores with cache sizes of 1 KB, 2 KB, 4 KB and 8 KB respectively. Then the following four applications were executed on each core: pi calculus (pi), the Jacobi-Richardson method to solve systems of linear equations (jacobi), and the two sorting methods bubble sort (bubble sort) and quick sort (quick sort). These programs were chosen since they have different characteristics with respect to execution time being determined largely by the CPU or largely by memory, or somewhere in between. The number of cache misses for each of these programs and cache sizes can be seen in Table 2. Since the four programs have their largest improvement at different cache size changes, de Abreu Silva et al. conclude that they can be classified into different classes, and thus can be scheduled efficiently. Thus when four different cache size are used (1 KB for pi, 2 KB for quick sort, 4 KB for jacobi and 8 KB for bubble sort, resulting in a total cache size of 15 KB), a lower miss rate can be obtained than when using 4 KB for each core, which is 16 KB in total. No exact results regarding energy consumption have been obtained. However, since the cache miss rate has dropped considerably in some cases, there is a high probability that energy consumption has dropped. More research has to be done to obtain concrete results.

B. Solution 2: Prefetching in Embedded Mobile Systems Can Be Energy-Efficient

While data prefetching has been very successful in high-performance computing systems, it has been found not to be sustainable in embedded systems, as it can increase energy consumption significantly. However, since embedded systems have become increasingly powerful, especially modern smartphones, Tang et al. [3] study the impact of data prefetching on energy consumption in mobile devices.
As modern mobile systems must execute a large variety of applications, three different sets of benchmarks have been tested: benchmarks which test XML data processing, multimedia and entertainment workloads and gaming and data mining applications. Also, six hardware prefetchers have been implemented, of which the details can be seen in Table 3.

First the cache has been simulated to study the impact on the performance. The results of this show that all six different prefetchers have similar performance, with on average an increase of 5% with respect to a standard configuration without prefetching. To study the impact on energy consumption, energy parameters have been modelled for both 90 nm and 32 nm technology. Simulation showed that in 90 nm technology all prefetchers cause a, sometimes significant, increase in energy, which suggests only conservative prefetchers can be energy-efficient in 90 nm technology. However, in 32 nm technology, some prefetchers show a decrease in energy consumption, with only a maximum increase of 25% in energy consumption for select cases.
Lastly, an analytical model is proposed to determine under which conditions prefetching can be energy efficient, as

well as to check which simulated prefetchers are energy efficient. A metric called the Energy Efficiency Indicator (EEI) was proposed, which is defined as the difference in performance gain and the ratio of the energy overhead that is incurred over the original static energy. This indicated that in the case of 90 nm technology only P4 is energy efficient with an EEI of 0.03. In the case of 32 nm technology, P1 had an EEI of 0.03 and P4 had an EEI of 0.05, which means both are energy efficient. From this study can be concluded that as technology advances, prefetching does not necessarily impact energy efficiency as heavily. In some cases, it seems that prefetching can even reduce energy consumption, although only slightly. Additionally, prefetching can be considered to be included in embedded mobile systems.

C. Solution 3: Topologically Homogeneous Power-Performance Heterogeneous Multicore Systems

In the continuous search for optimising the hardware architecture of a processor for lower power consumption with less or no performance degradation, Chakraborty and Roy [4] propose the idea of designing a processor with multiple homogeneous cores with different VF characteristics, thereby creating a heterogeneous architecture meant to optimise the power-performance characteristic, while taking thermal constraints into account. The motivation behind this approach lies in the fact that the mostly used DVFS technique causes much degradation in energy efficiency when the nominal operating frequency is lowered with respect to ground-up designs for this nominal frequency, such as the one discussed in the paper. Furthermore, the building blocks of this processor are homogeneous, as can be seen in Figure 4. This means that the processor is built up using identical blocks (clusters) resulting in low-costs benefits, because this block only has to be designed once resulting in less development costs.

The whole design essentially consists of two steps described below.

1) VF domain selection: At this stage, the VF domains which are most energy efficient for the workload execution (consisting of multiple different applications) on the specific amount of cores with given hardware technology are chosen according to a statistical analysis. This can be done via an exhaustive search, but this problem would become impractical. Therefore, for each tuple (V,F) the energy as a function of the frequency is calculated (called O). The next step is to calculate the probability (called Y) that a VF domain is able to produce a relatively high enough IPC with respect to the base case with a varying workload. The performance is thus measured in IPC, where the base case (IPC_{base}) stands for the IPC when an application is run on a core with the highest possible VF combination (resulting in the highest power consumption). In order to obtain the optimal solution, a Monte Carlo simulation is performed where from 1000 randomly chosen applications the lowest O is selected, which still satisfies the

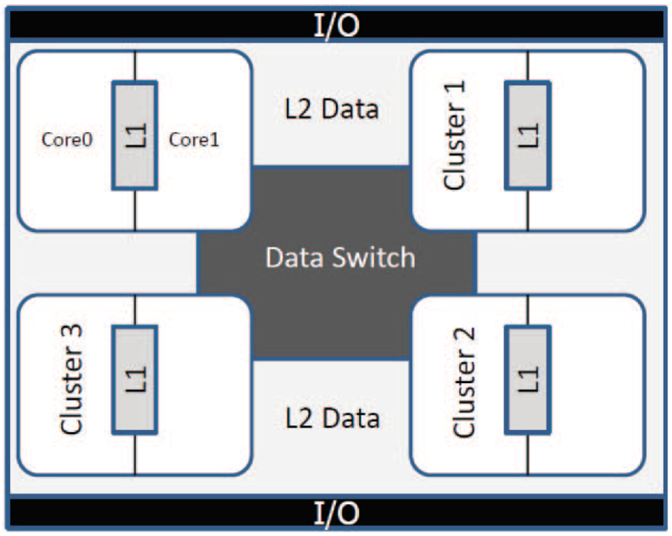

Fig. 4. Clustered multi-core system [4]

criterion of Y being above a certain predefined threshold (to ensure enough performance).

2) VF domain optimisation: After having assigned a VF domain to each core, this core is optimised for the determined frequency by adjusting the components for different voltages until the right power-performance balance has been found. Important here is to take task migration (as described in the next section) into account.

3) Task migration: Besides the two design steps in order to find the optimal VF domain selection and optimisation as described above, another aspect has to be considered: task migration. By using task migration, a task can be moved to another core/cluster to be further executed in case that is desirable from an throughput as well as from an energy efficiency point of view. Task migration can be performed within the same core or between different clusters. Inter-core task migration causes less hardware overhead than inter-cluster task migration. Therefore, inter-core reassignment of tasks is preferable to inter-cluster reassignment.

4) Experimental results: To conclude, this paper tries to improve the power-performance characteristic of a heterogeneous multi-core processor with low-costs benefits by reusing multiple homogeneous cores and assigning different VF domains to them. It turns out that the new hardware architecture (containing different VF domains) is 11-22% more energy efficient (and also has a higher throughput) with respect to the tested architectures using DVFS. This improvement is due to the intelligent task assignment to the different cores, which ensures fast enough execution, but also a higher energy efficiency, which is caused by the optimal VF domain selection.

D. Solution 4: Energy-Efficient Reconfigurable Cache Architectures for Accelerator-Enabled Embedded Systems

In many high-performance embedded systems, the processor is coupled to more specialised accelerators, to improve performance and energy efficiency. However, these accelerators share the same memory with the processor, but show different memory behaviour, due to high parallelism in execution. To improve this potential bottleneck, Fermahini-Farahani, Sung Kim and Morrow [5] propose the use of configurable L1 data (L1D) cache. The configurable L1D cache can have two forms (see Figure 5): 1) the processor and accelerator use the same L1D cache in total and 2) the processor and accelerator will have their own equal part of the L1D cache. By using a state-space exploration on a set of programs, a selection block can be added to the L1D cache which will choose the optimal configuration for each application. This results in an improvement up to 64% in energy efficiency. However, during the state-space exploration also the effect of the cache capacity, the number of read/write ports and the degree of set associativity was taken into consideration. This means that when configuration 1 is chosen for two different programs, both programs could be executed using configuration 1 (the processor and accelerator use the same L1D cache in total) with a different configurations (e.g. cache capacity) for the shared L1D cache. If a designer only wants to use the given solution so far, a fixed cache configuration can be chosen, which will lead to the "best" overall performance (for a fixed cache). However, the paper continues by giving a method to get the correct configuration regarding these parameters, by using configurable ports, this provides a configurable trade-off between cache capacity (size in kB) and cache bandwidth (amount of ports). By looking at each program, to see whether it favours more bandwidth or capacity, the optimal balance per program can be analysed. When the program is recognised, the correct configuration is chosen, and can lead to an improvement in energy efficiency up to 33%. In the paper the energy efficiency is defined as the energy-delay product (EDP).

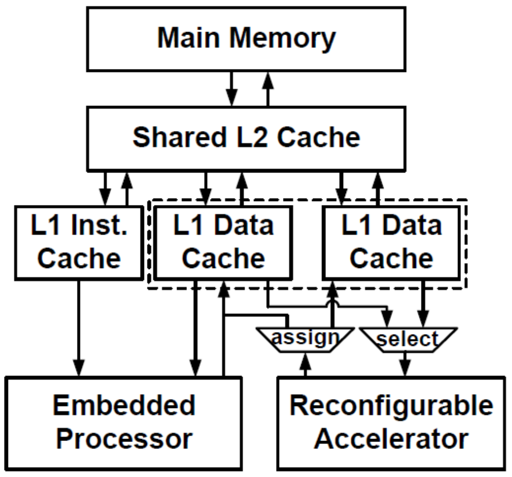

Fig. 5. High-Level structure of proposed reconfigurable L1D cache [5]

III. COMPARISON

While all solutions either show that energy consumption has decreased or show promising results which may imply an increase in energy efficiency, not all solutions are equally effective or applicable in all situations. The different solutions will mainly be compared with respect to the effectiveness and usability. The main concern of the first solution is that no concrete results regarding energy consumption have been obtained, which means no definite answer on whether it can increase energy efficiency. However, better cache performance may indicate a lower energy consumption.
While the second solution shows an increase in energy efficiency, it is low compared to the other three solutions. This might improve in the future, however, as transistor technology improves. Additionally, the effectiveness of this solution is largely dependent on the type of program, and thus also the system. Furthermore, although the results show a higher energy efficiency in some cases, this solution mainly focuses on whether prefetching can be energy efficient, not if it can be used to decrease energy consumption. This means that potentially better energy efficiency could have been achieved over the base case if that was the goal of the study. Currently, this solution is less effective when compared to the others.

The third solution indicates a good reduction of 11-22 % in energy consumption, and additionally, as each block needs to be designed only once, the designing costs are low. However, the system needs to be partially designed for different applications, meaning more design costs. This means that each application will need its own designed processor, meaning less reusability.

Finally, the last solution shows improvements up to 33%, when the correct configuration is used each time. However, this is only useful in higher performance systems, as this solution is only applicable when accelerators are used. Additionally, the most optimal configuration for each program can only be chosen reliably if it is already known, which means it will have to be tested beforehand. This means either the system can only execute a limited amount of predefined programs efficiently, or compromises need to be made in which the optimal configuration is somehow predicted. While this argument can be made for the first and third solutions as well, this impacts the last solution more heavily.
Another disadvantage concerning the first and third solution can be indicated. These two solutions both rely upon the fact that there are four or potentially more cores available with different configurations. This will generally only be the case in high performance systems, and can thus also not be generally applied.

IV. Conclusions

Four solutions are discussed for optimising the energy efficiency of heterogeneous multi-core embedded systems. First different cache sizes are used to reduce the miss rate. However, this solution has not shown results on the impact of energy consumption. The second solution uses prefetching to make the system more energy efficient. This method works best on smaller technology, but still only marginally increases the efficiency of some programs (on average 5%). The third solution uses pre-designed cores with the optimal VF settings for pre-defined applications and increases efficiency by 11-22%. This solution is limited by the pre-defined applications and the quality of the scheduler. The final solution, uses reconfigurable cache architectures and show an increase in energy efficiency up to 64%, however the solution works best when the cache can be configured by trading-off cache size with bandwidth and this can increase the energy efficiency up to 33%. However, this solution needs prior information to be optimal and can only be used in high-performance embedded systems. By inspecting all the solutions, a potential to increase energy efficiency is shown. The solutions should be chosen dependent on the context in which the embedded system will perform (e.g. the programs are known beforehand, system allows for accelerators, etc.). These dependencies are caused by the space in which heterogeneous multi-core embedded systems operate. In this space, certain subspaces can be identified to optimise, like the embedded systems with accelerators. This means that for this paper, optimising multiple subspaces is a valid approach to get a coverage for optimising all sorts of heterogeneous multi-core embedded systems. For future research, the impact on reducing cache misses on the energy efficiency can be analysed, due to the missing results in the analysed literature for this paper. Furthermore, the use of reconfigurable cores can be analysed. [6] This can be interesting, because the cores can be set into an optimised configuration before the program starts, which can result in higher energy efficiency and performance. Moreover, the composition of cores can be analysed to see if different configurations can increase energy efficiency. This is also based on the principle of reconfiguring cores, only the cores will now be configured in the design stage of the embedded system to perform optimal for the expected programs, instead of being reconfigurable during runtime.

References

[1] Peter Puschner, *Embedded systems for safety-critical and mixed-criticality applications*,
P. Puschner, Embedded systems for safety-critical and mixed-criticality applications, in *2013 2nd Mediterranean Conference on Embedded Computing (MECO)*, June 2013, pp. 12.

[2] Bruno de Abreu Silva, Lucas Albers Cuminato and Vanderlei Bonato, *Reducing the Overall Chache Miss Rate Using Different Cache Sizes for Heterogenerous Multi-Core Processors*,
B. de Abreu Silva, L.A. Cuminato and V. Bonato, "Reducing the Overall Chache Miss Rate Using Different Cache Sizes for Heterogenerous Multi-Core Processors", in *2012 International Conference on Reconfigurable Computing and FPGAs*, Mexico, December 2012.
doi:10.1109/ReConFig.2012.6416783

[3] Jie Tang, Shaoshan Liu, Zhimin Gu, Chen Liu and Jean-Luc Gaudiot, *Prefetching in Embedded Mobile Systems Can Be Energy-Efficient*,
J. Tang, S. Liu, Z. Gu, C. Liu and J. Gaudiot, "Prefetching in Embedded Mobile Systems Can Be Energy-Efficient", in *IEEE Computer Architecture Letters*, Vol. 10, No. 1, January-June 2011, pp. 8 - 11,
doi: 10.1109/L-CA.2011.2

[4] Koushik Chakraborty and Sanghamitra Roy, *Topologically homogeneous power-performance heterogeneous multicore systems*,
K. Chakraborty and S. Roy, Topologically homogeneous power-performance heterogeneous multicore systems, in *2011 Design, Automation Test in Europe*, March 2011, pp. 1-6.

[5] Amin Farmahini-Farahani, Nam Sung Kim and Katherine Morrow, *Energy-efficient reconfigurable cache architectures for accelerator-enabled embedded systems*,
A. Farmahini-Farahani, N. S. Kim and K. Morrow, "Energy-efficient reconfigurable cache architectures for accelerator-enabled embedded systems," *2014 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, CA, USA, 2014, pp. 211-220.
doi:10.1109/ISPASS.2014.6844485

[6] Stephan Wong, Thijs van As and Geoffrey Brown, *-VEX: A reconfigurable and extensible softcore VLIW processor,*
S. Wong, T. van As and G. Brown, "-VEX: A reconfigurable and extensible softcore VLIW processor," *2008 International Conference on Field-Programmable Technology, Taipei*, 2008, pp. 369-372.
doi: 10.1109/FPT.2008.4762420