NTNU – Trondheim
Norwegian University of
Science and Technology

Energy Efficiency Experiments on Mali Powered Exynos 5 using OmpSs

Rune Holmgren

December 2014

PILOT PROJECT FOR MASTER THESIS

Department of Computer and Information Science

Norwegian University of Science and Technology

Supervisor 1: Professor Lasse Natvig

Co supervisor: Antonio Garcia Guirado

Problem statement

Here is the problem statement.

Acknowledgements

Here are the acknowledgements.

Abstract

This is the abstract.

Contents

Chapter 1

Introduction

1.1 Motivation

Increase of performance and power efficiency are the main goal of processor designers. Unfortunately we are currently reaching the limits of the current strategies for further development. For some time, our processors have been struggling to achieve increased performance. Heat stops us from driving the clock frequency higher, while memory is lagging more and more behind. A solution to enable continued performance growth is multi core processors, and for the last decade this has been the focus. Unfortunately adding cores will not be a sustainable solution forever. As the amount of cores grow, they are still competing for the same system resources and may have to wait for each other to complete calculations on data with dependencies.

A promising solution to this issue is heterogeneous multi-processor systems. Heterogeneous multi-processor systems utilize multiple different processor cores in the same system. This allow different parts of a program to be executed on a suitable processor. By using a suitable core for each part of the program it is possible to achieve better performance than homogeneous multi-processor systems.

1.2 Project Scope and Goal

This pilot projects main goal is to do preliminary research and experiments on the energy efficiency of the Exynos 5 processor, with the intent to use the results next spring in my master thesis. The goal of this research is to explore the potential of the task based programming model heterogeneous multi-processor systems.

1.3 Problem Statement Interpretation and Approach

- Task 1: Implement or adapt suitable experiment applications for testing energy efficiency.
- Task 2: Implement some energy efficiency measurement application for both Arndale duo and ODROID-XU3.
- Task 3: Optimize experiment applications for both platforms.
- Task 4: Gather performance and energy efficiency results from the experiment applications on both platforms.
- Task 5: Analyze and evaluate experiment results.

TODO: Introduce how these tasks were solved.

1.4 Outline

TODO: This section need to be completed after the outline of the report is done.

Chapter 2

Related work

Chapter 3

Background

3.1 Energy measurement

3.2 NEON

NEON is a general-purpose single input multiple data (SIMD) technology implemented in the ARM Cortex A series of processors. It is able to run SIMD instructions on 128bit registers. By utilizing the NEON unit of the ARM processors, it is possible to achieve parallelism in each separate core. This will often open for great performance boost on problems like the ones explored in this paper. Each register may be filled with single precision floating point numbers ranging from 8 to 64 bit each. In future generations of the ARM ISA there will be support for other data types as well. Different implementations of NEON exist in the Cortex A cores, and while the even the simple implementations in smaller cores like the A7 can give great performance boost, the implementations present in the newest cores are performing even better. The A15 offer two NEON units, and the instruction pipeline to start the cores are shorter than in simpler implementations.

3.3 Task based programming

Task based programming allow a programmer to work with parallel programs, with an abstraction from the parallelization itself. When programming with this model, the program can be split into tasks which can run in parallel. When the program run, it will run a task manager as part of the program. This task manager can dynamically assign tasks to the processors, and the programmer does not have to handle all the time consuming tasks related to manual parallelization. As long as the programmer correctly handle dependencies in the parallelized code, it will be possible to write this kind of code as if it was serial.

The task based programming model also allow simpler development of portable programs. When the program is running tasks on available CPUs, it is not a problem to allow it to run on larger or smaller numbers of processors, and even clusters can support the program. This model even allow the tasks to run on different types of processors in a heterogeneous environment.

3.4 OpenMP Super scalar

OpenMP Super scalar (OmpSs) is a extension of the OpenMP API to integrate features from the StarSs programming model. It is currently under development at the Barcelona Supercomputing Center. The goal of OmpSs is to extend the programming model to support a wide range of processors. The OmpSs programming model will run on a wide variety of different systems, such as traditional personal computers, clusters, shared memory systems and heterogeneous processors. While the software is not yet completed or fully tested, there have been several reports exploring it's potential. The results have proven OmpSs as an efficient solution on both clusters and heterogeneous systems utilizing OpenCL and CUDA.

3.5 Heterogeneous multi-processor

Heterogeneous multi-processor systems have multiple different processors, opposed to traditional multi-processor systems. A typical modern processor have several processors, and a program can run effectively by having threads running parts of thesis work on each of them. This work is often of such a nature that it can run better on a different processor. Sometimes it can run just as well on multiple simple processor, while using less die space and energy. In other instances, an advanced processor with some special capabilities, like vector instructions, can be more efficient.

This kind of processors have a potential to help us overcome the challenges that are emerging in processor development. Unfortunately they also introduce several new challenges.

3.6 Experiment platforms

3.6.1 Arndale Board

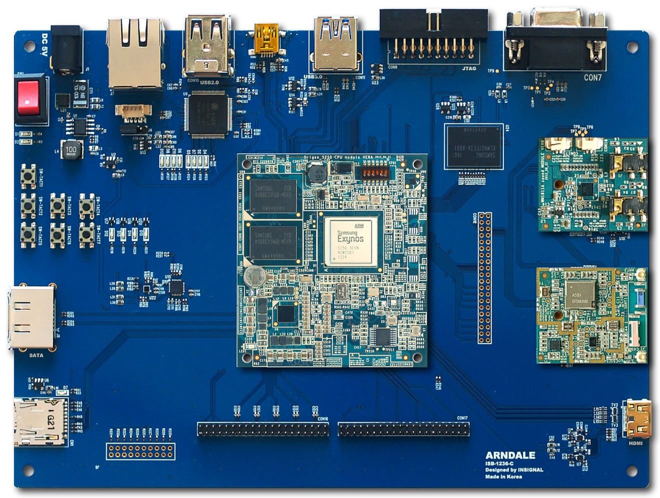

Figure 3.1: Arndale Duo

The Arndale Duo is a computing system mounted on a single board. It is fitted with an Exynos 5250 SoC, which contain a dual core Arm Cortex-A15 , as well as an ARM Mali T-604 GPU. This computer offer a range of supported Linux distributions, as well as the OmpSs programming model. The computer was used in the 2014 master thesis "Acceleration with OmpSs and Neon/OpenCL on ARM Processor" by Trond Inge Lillesand. The thesis lay a lot of the ground for this pilot project and planned master thesis.

3.6.2 ODROID-XU3

The ODROID-XU3 is a new single-board computing system, offering interesting properties for these experiments. The system has an Exynos 5422 heterogeneous Soc. Exynos 5422 has a quad core ARM Cortex-A15 CPU and a ARM Mali T-628 GPU, but also a smaller quad core ARM Cortex-

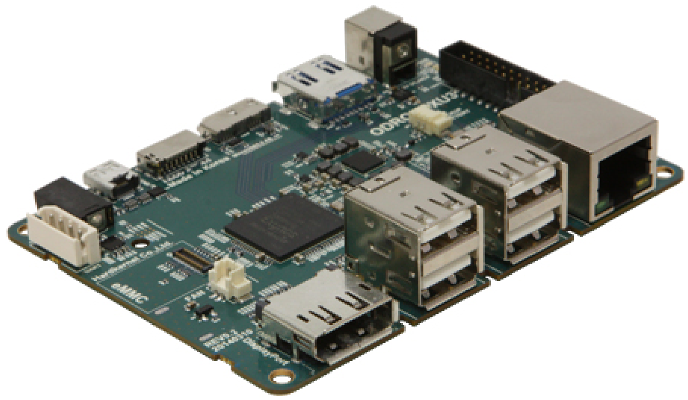

Figure 3.2: ODROID-XU3

A7 processor. These 3 different processing units can be used simultaneously to solve problems. In this paper, and the planned master thesis following it, the potency of this kind of heterogeneous processor will be explored.

Power monitoring

The ODROID-XU3 comes with integrated power monitoring tools. Implemented in hardware, it have got 4 current sensors sitting on the power pins of the Exynos SoC. The energy monitors are indicated in figure 3.3. These monitor the current going through the large CPU cores, the small CPU cores, memory and GPU respectively. In addition to the current sensors, the power management for the SoC is also available to the programmer, making supply voltage to the components known. By using the voltage and current, the power consumption is known. As a whole, the system offer frequency, voltage, current, temperature and power readings in real time. These fine energy and performance metrics make the system highly suitable for developers. They are able to run their programs, collect energy profiling data and optimize their software based on the result.

Performance

The ODROID-XU3 comes with the ARM Cortex-A15 limited to 2GHz and the ARM Cortex-A7 limited to 1.4G Hz. There is 2GB of memory available running at 800 MHz, and the ARM Mali T-628 GPU run at 600 MHz. This performance place it at the higher end of SoCs, but not quite in the top, as it is beaten by systems like TODO A reference system here). The ODROID-XU3

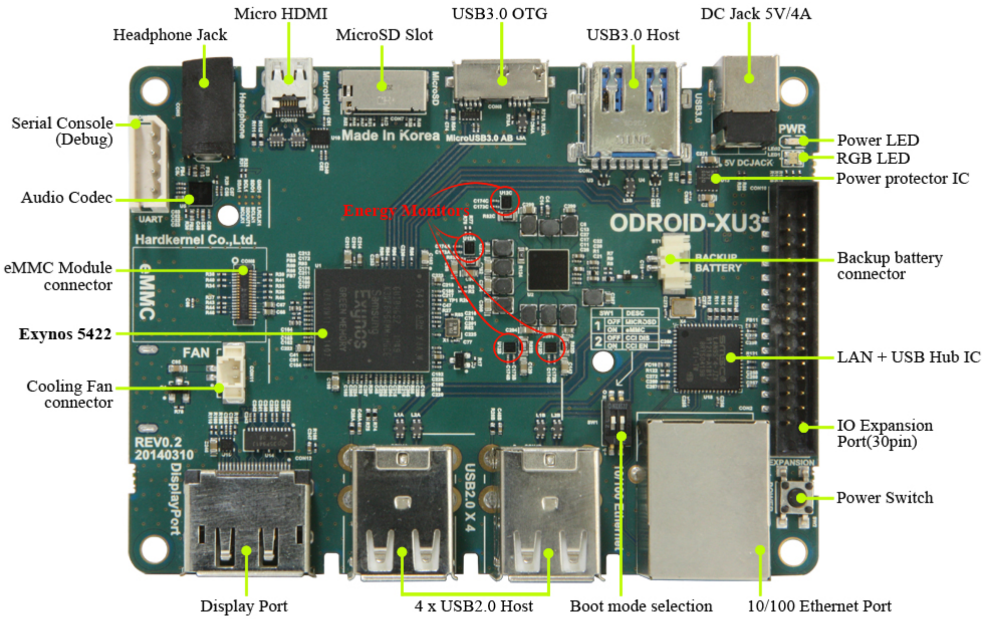

Figure 3.3: ODROID-XU3 annotated (hardkernel.com[1])

feature the new eMMC 5.0 standard for storage. The performance of this standard outperform both older eMMC standards, as well as memory card readers, which other similar systems may contain. The ODROID-XU3 can achieve a read/write performance of 198/74 MB/s[1], which a lot better than older systems like the Arndale Duo.

In addition to the eight processor cores, the system also feature a ARM Mali T-628. The Mali T-628 function both as a regular GPU, as well as a GPGPU. It support both openGL and DirectX, and is able to produce graphics for all but the most demanding gaming and simulation purposes. In addition to this it is able to run computations with Open CL. This mean that problems with parallel parts, can be solved efficiently utilizing the GPU.

3.6.3 ARM Cortex-A15

Performance	1.0 GHz to 2.5GHz
L1 Cache	64KB
L2 Cache	4 MB
L3 Cache	None in core, may be implemented shared in multi core system.
Architecture	ARMv7-A
Architecture	ARMv7-A
	TrustZone® security technology
	NEON™ Advanced SIMD
	DSP & SIMD extensions
	VFPv4 Floating point
	Hardware vitalization support
	Integer Divide
	Fused MAC
	Hypervisor debug instructions
Memory management	40-bit ARMv7 Memory Management Unit

3.6.4 ARM Cortex-A7

The ARM Cortex-A7 is designed to be a low power alternative to the ARM Cortex-A15 and ARM Cortex-A17, with the same supported ISA and features. This enable the ARM Cortex to be paired with it's larger relatives in a ARM big.LITTLE configuration.

Performance	1.2 GHz to 1.6GHz
L1 Cache	8-64KB
L2 Cache	up to 1 MB
L3 Cache	None in core, may be implemented shared in multi core system.
Architecture	ARMv7-A
Supported features	ARM Thumb-2
	TrustZone® security technology
	NEON™ Advanced SIMD
	DSP & SIMD extensions
	VFPv4 Floating point
	Hardware vitalization support
	Integer Divide
	Fused MAC
	Hypervisor debug instructions
Memory management	40-bit ARMv7 Memory Management Unit

3.6.5 ARM Mali T604

Performance	533 MHz
	17 GFLOPS
Multi core support	1-4 cores
API Support	OpenGL 1.1, 2.0, 3.0 and 3.1
	OpenCL 1.1
	DirectX 11
	RenderScript
Anti-Aliasing	4xFSAA with minimal performance drop
	16xFSAA
Cache	32-256KB L2 cache

3.6.6 ARM Mali T628

Performance	533/695 MHz
	17/23.7 GFLOPS
Multi core support	1-8 cores
API Support	OpenGL 1.1, 2.0, 3.0 and 3.1
	OpenCL 1.1
	DirectX 11
	RenderScript
Anti-Aliasing	4xFSAA with minimal performance drop
	16xFSAA
Cache	32-256KB L2 cache

3.7 Algorithms

Here I will write about the algorithms used in the experiments.

Chapter 4

Setup and Methodology

4.1 Test platforms

4.1.1 Arndale Duo

The Arndale Duo board was used for some preliminary research in this thesis. It was chosen because we had experience from earlier student projects using this board. It's feature set and properties are elaborated in section **??** **??**.

4.1.2 ODROID-XU3

The ODROID-XU3 single board computing system was used for most of the experiments of this thesis. Fitted with an Samsung Exynos 5422 SoC it offer the heterogeneous properties that will be explored in detail in the planned master thesis. In addition, the board offer multiple energy monitors, enabling precise data gathering. It's feature set and properties are elaborated in section 3.6.2 ODROID-XU3.

SoC	Samsung Exynos 5250
CPU	
Model	ARM Cortex-A15
Manufacturing process	32nm
Maximum clock frequency	1.7GHz
Number of cores	2
L2 Cache	1MB
L1 Cache	32KB
GPU	
Model	ARM Mali-T604
Maximum clock frequency	600 MHz
Number of cores	4
Memory	
Available memory	2 GB
Maximum clock frequency	800MHz
Operating system	
Distribution	Linux Ubuntu
Version	TODO

Table 4.1: Arndale Duo Specifications

4.2 Software

Software	**Version**
Nanos++	0.9a
Mercurium	1.99.4
Extrae	3.0.1
gcc	4.8.2

Table 4.3: Third party software and frameworks used in the experiments.

4.3 Compilation and running of test benches

Mention frequency scaling here.

4.4 Performance measurement

In the experiments being run in this pilot project, execution time was used as the primary metric for performance. While running the experiments, the POSIX function gettimeofday() is used

SoC	Samsung Exynos 5422
CPU 1	
Model	ARM Cortex-A15
Manufacturing process	32nm
Maximum clock frequency	2.0GHz
Number of cores	4
L2 Cache	512KB
L1 Cache	32KB/32KB I/D
CPU 2	
Model	ARM Cortex-A7
Manufacturing process	32nm
Maximum clock frequency	1.4GHz
Number of cores	4
L2 Cache	2MB
L1 Cache	32KB/32KB I/D
GPU	
Model	ARM Mali-T628 MP6
Maximum clock frequency	600 MHz
Number of cores	4
Memory	
Available memory	2 GB
Maximum clock frequency	933MHz
Operating system	
Distribution	Linux odroid
Version	3.10.54+

Table 4.2: ODROID-XU3 Specifications

before and after running the main part of the application. The time difference is a used as a measurement of how good the performance is.

In the planned master thesis, PAPI (Performance API) may be used to measure more detailed aspects of the performance. PAPI is able to access a lot of different performance measurements from systems. The available metrics include among others; Cache utilization and hit rate, memory utilization and bus utilization.

4.5 Energy efficiency measurement

Because of the different features of the two experiment platforms, two different energy efficiency measurement schemes was used.

4.5.1 ODROID-XU3

As elaborated in section 3.6.2 ODROID-XU3, the ODROID-XU3 feature 4 current sensors. These current sensors are used to monitor the current flow to the large CPU, small CPU, GPU and memory respectively. In addition to the current sensors, it is possible to read the supply voltage to each of these components. Based on these two sensor readings, we can calculate the power consumption of each of the components in real time while running our applications. Using this scheme for power measurement, we are able to get readings with high precision of each component. The results will show how the consumption vary with different application configurations both for the application as a whole and for different stages of execution.

$Power\ consumption(Watt) = Current(Ampere) \times Voltage(Volt)$

4.5.2 Arndale Duo

Chapter 5

Implementation

Chapter 6

Result and Discussion

In this chapter the results of the experiments described in earlier chapters are presented. The chapter is divided into three sections, covering the 3 different types experiments that were run. First the experimental optimization towards our systems properties are presented. Then the experiment results regarding system performance will be presented. Finally the results regarding the systems energy efficiency are presented.

6.1 Optimization

Different systems may perform better with different programs. To ensure that the results we find are represented by programs well suited for the system, the experiments were tested with different degrees of loop unrolling. By unrolling loop iterations, the task sizes increase. This reduce the overhead of initiating iterations. It also reduce overhead related to loop control, like end of loop tests and loading data into new memory locations necessary for each loop iteration. There is however a limit to how much unrolling can be done. At some point the size of the task will create difficulties like early cache eviction of data. The appropriate amount of cache may vary from system to system. Because of this, experiments were run with different unroll degrees, and on all the processor configurations that will be used for later experiments.

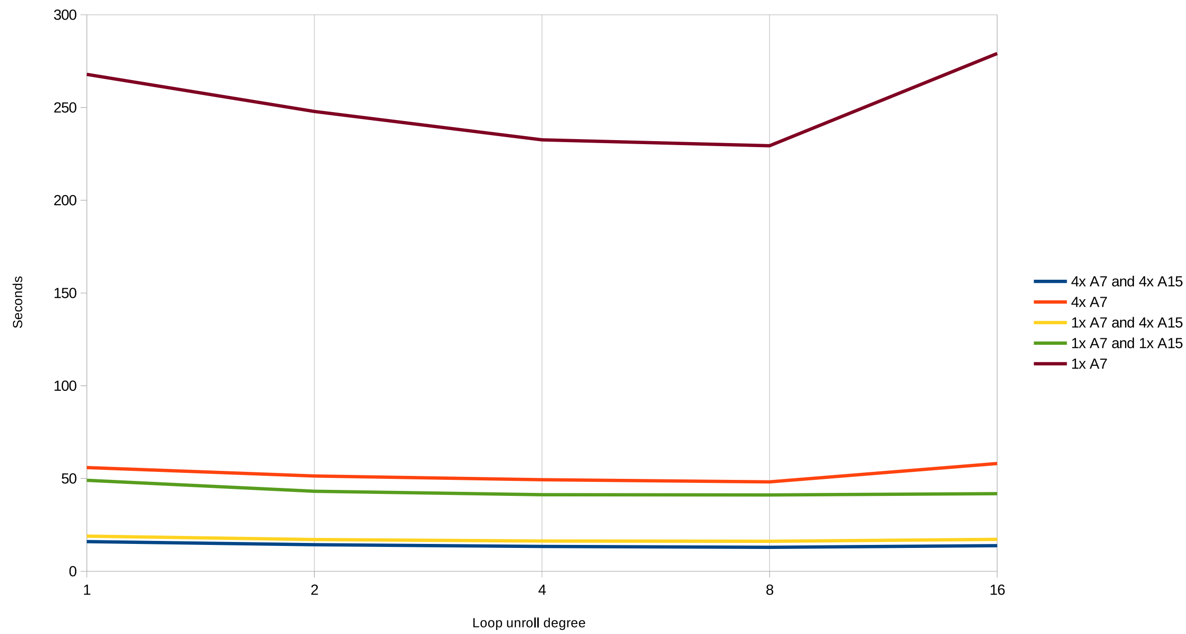

Figure 6.1: Execution time of 2D-Convolution with different degrees of loop unrolling running on different processor configurations.

Processor configuration	Loop unroll degree				
	1	2	4	8	16
4x Cortex-A7 and 4x Cortex-A15	15.9875	14.3006	13.3692	12.8891	13.8013
4x Cortex-A7	55.8885	51.3407	49.3267	48.1665	58.0834
1x Cortex-A7 and 4x Cortex-A15	18.9248	17.082	16.279	16.1658	17.1759
1x Cortex-A7 and 1x Cortex-A15	49.0195	43.1061	41.2549	41.1286	41.8136
1x Cortex-A7	267.8882	247.8783	232.5777	229.3884	279.1135

Table 6.1: Execution time of 2D-Convolution with different degrees of loop unrolling on different processor configurations.

The results show that a loop unroll degree of 8 was optimal for this specific implementation of 2D-Convolution on ODROID-XU3. This was observed across the results with all tested processor configurations. Because of this result, the remaining experiments are all run with a loop unroll degree of 8.

As mentioned loop unrolling is limited. In 2D-Convolution there are many read write operations in the loop body. As the size of this loop body increase, the amount of data accessed by each iteration grow. Eventually this data does no longer fit in cache. There is reason to suspect that this is what we are observing in the performance loss at loop unroll degree 16.

6.2 Performance

Even with energy efficiency as the focus, performance data are still interesting. The data are both useful on their own to observe the power of the system, as well as being a way to compare energy results. When energy efficiency of a system is measured, it is important to look at the energy data of the system in light of it's computational power. A system consuming low amounts of power is not as impressive if it is equally low performing. These are the results of running 2D-Convolution with 8 as the loop unroll on different processor configurations.

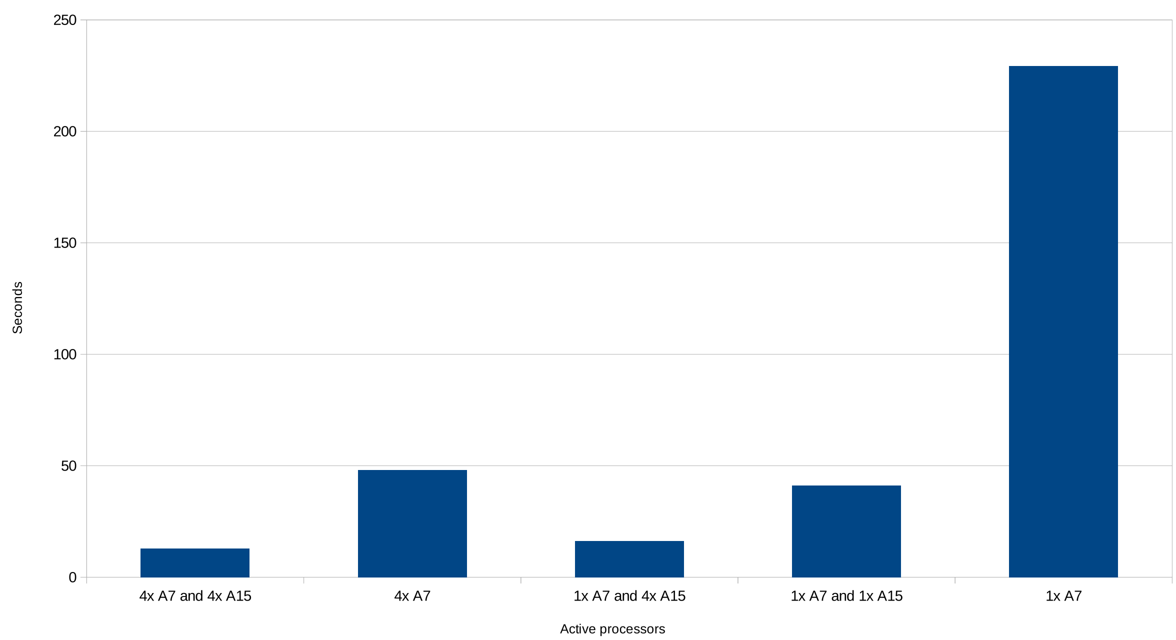

Figure 6.2: Execution time of 2D-Convolution on different processor configurations.

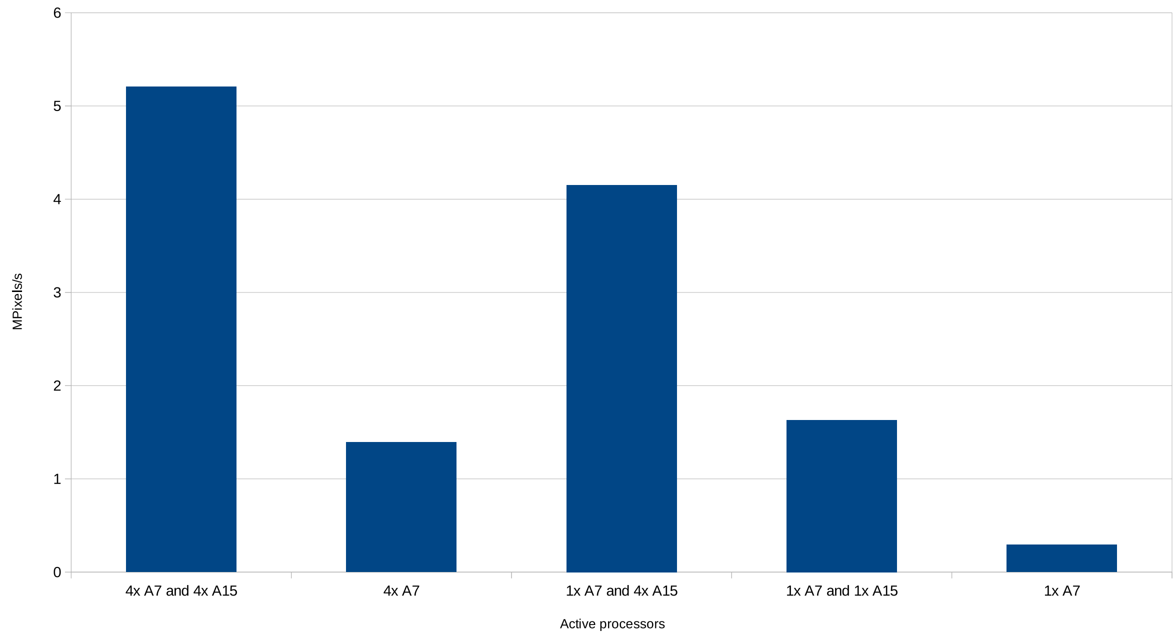

Figure 6.3: MPixels/s of 2D-Convolution running on different processor configurations.

Processor configuration	Execution time (s)	Performance (MPixels/s)
4x Cortex-A7 and 4x Cortex-A15	12.8891	5.2066
4x Cortex-A7	48.1665	1.3932
1x Cortex-A7 and 4x Cortex-A15	16.1658	4.1512
1x Cortex-A7 and 1x Cortex-A15	41.1286	1.6316
1x Cortex-A7	229.3884	0.2925

Table 6.2: Performance of 2D-Convolution with different processor configurations.

4 x 1 A7 is worse than 1 x 4 A7 4 X 1 stk A7 + 1 stk A15 is better than 4 stk A7 and 4stk A15 All processors perform 3.7 times better than only the small cores.

6.3 Power measurements

As described in chapter 4 Setup and Methodology, separate energy measurements for the different SoC components were gathered during execution of the experiments. The each of the

4 components value was logged every 200 ms. In figure 6.4 you can see the raw energy measurements for a single execution of 2D-Convolution running on all 8 processors. We can here observe the energy consumption of the program throughout the execution for different components. The total energy consumption can be calculated from the data. But observations regarding power usage during different execution stages can also be analyzed. The same kind of data were gathered for a range of different processor configurations.

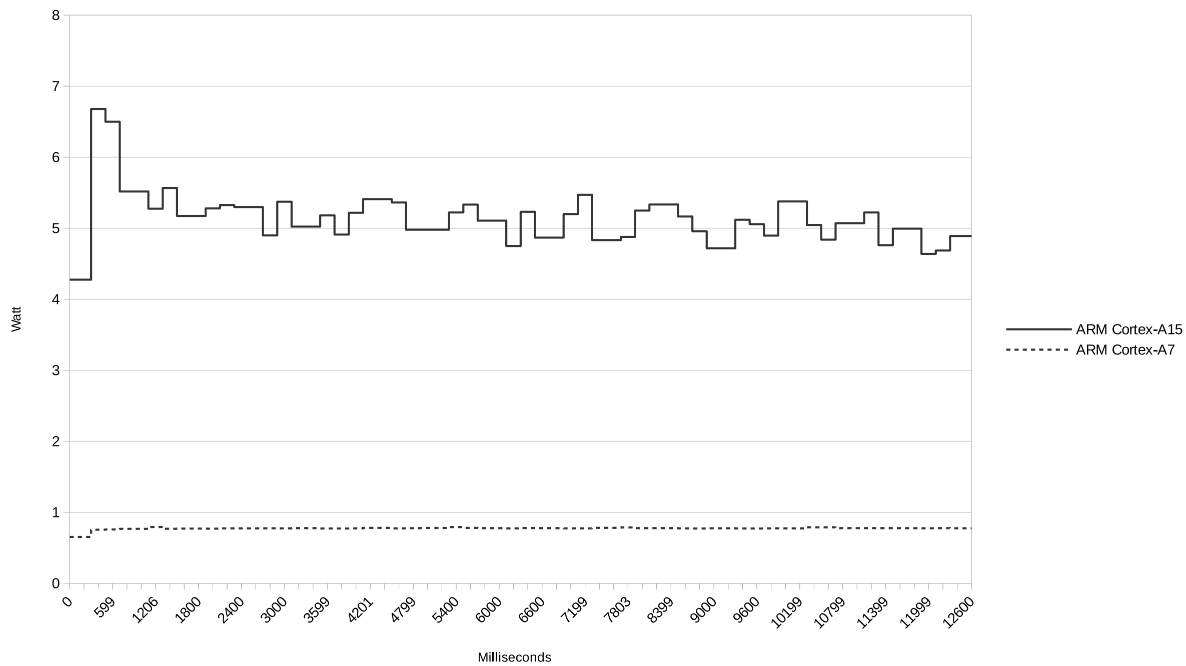

Figure 6.4: Power consumption for the large and small cores during execution of 2D-Convolution with unroll 8x. This is a full execution running on all 8 processors.

There is a spike in power consumption on both the small and large cores at the start of the execution. This spike can be caused by a lot of different reasons. There is not enough data to know what caused it. Typical reasons for such spikes are intense operations during initiation of the program, or simply hardware implementations causing a power surge when processor cores are suddenly powered from idle state. For the rest of the execution there are variations in power consumption, but the consumption vary around the same general level.

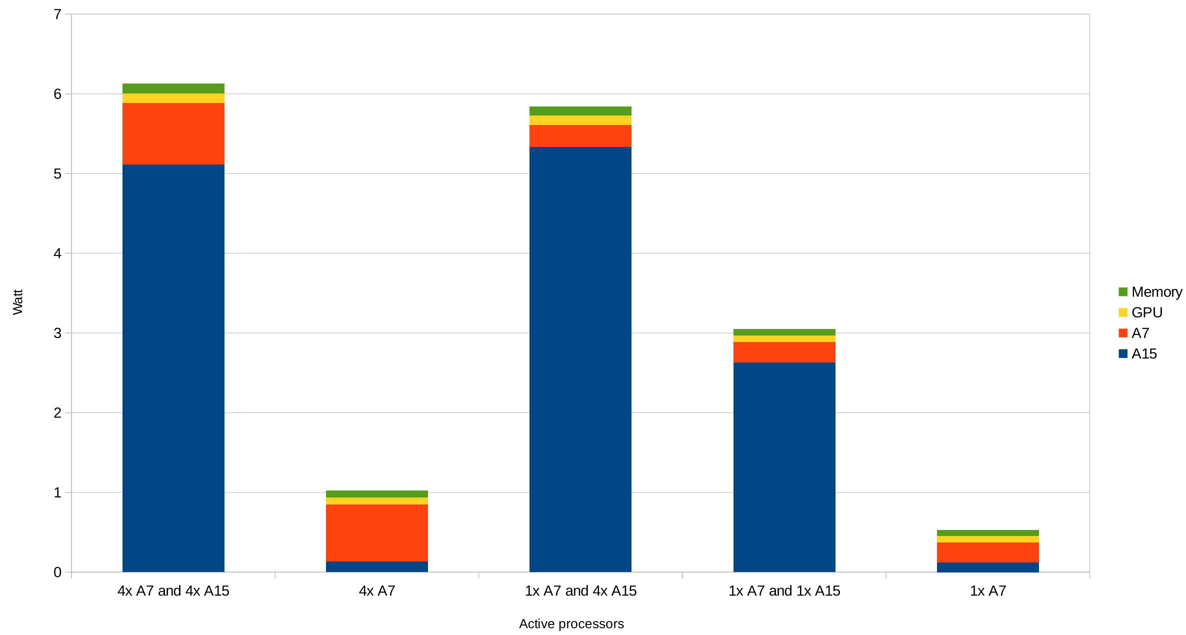

Figure 6.5: Average power consumed per second for different processors configurations running 2D-Convolution. The colors indicate different components, and the full height of each column is the accumulated total for the whole system.

Processor configuration	Power consumption per second for each component				
	Cortex-A15	Cortex-A7	Mali-T628	Memory	Total
4x Cortex-A7 and 4x Cortex-A15	5.1156	0.7693	0.1206	0.1208	6.1264
4x Cortex-A7	0.1362	0.7157	0.0899	0.0817	1.0238
1x Cortex-A7 and 4x Cortex-A15	5.3356	0.2758	0.1161	0.1098	5.8373
1x Cortex-A7 and 1x Cortex-A15	2.6341	0.2540	0.0818	0.0755	3.0456
1x Cortex-A7	0.1197	0.2535	0.0813	0.0709	0.5256

Table 6.3: Execution time of 2D-Convolution with different degrees of loop unrolling on different processor configurations.

Using the data gathered for each component in different processor configurations we can examine their efficiency. These data are presented in figure 6.5.

As expected we see that the power of the ARM Cortex-A7s is a lot lower than the power of the

ARM Cortex-A15. Turning off the power hungry Cortex-A15s reduce the total power to a sixth (0.17). This is useful for applications running that do not require high performance, but want to consume low power. This way of power saving can be increased even more by utilizing only a single Cortex-A7. This configuration is have the lowest power consumption per second, and is useful for standing by, or executing minor calculations while standing by. Running on only a single Cortex-A7 reduce the power of the system to a tenth (0.10) of that of the fully powered 8 core system.

In addition to the total power of the system, the results also allow us to analyze the power of each component. Turning off 3 Cortex-A15 cores only reduce the power to about half (0.51). Even turning all the Cortex-A15 cores off still leave some power consumed by them. Because of this power, running the system with only four Cortex-A7 cores still spend 13% of the total energy on the Cortex-A15 cores, which are not doing any work. Similarly turning off 3 Cortex-A7 cores only reduce power to a bit more than two thirds (0.35). These results reveal that the energy for each set of cores are not spent solely on powering the cores themselves, but also some peripheral components related to the cores. These peripheral components can not be turned partially of, and will consume some power when at least one core is on. They even consume some power when the cores are all off. Because of this overhead power from partially powered processors, there rarely occur situations where this is optimal. The results for such processor configurations will be analyzed further in this thesis to observe whether or not this is the case here.

6.4 Energy consumption measurements

A The power usage of the different system components is interesting. There is however many cases where the system can be put to another task, or shut down, when it is finished. In such cases it is interesting to observe the total amount of energy consumed completing the task. In figure 6.6 the data for each component power usage is multiplied with the execution time. The data height of each column is the amount of energy consumed.

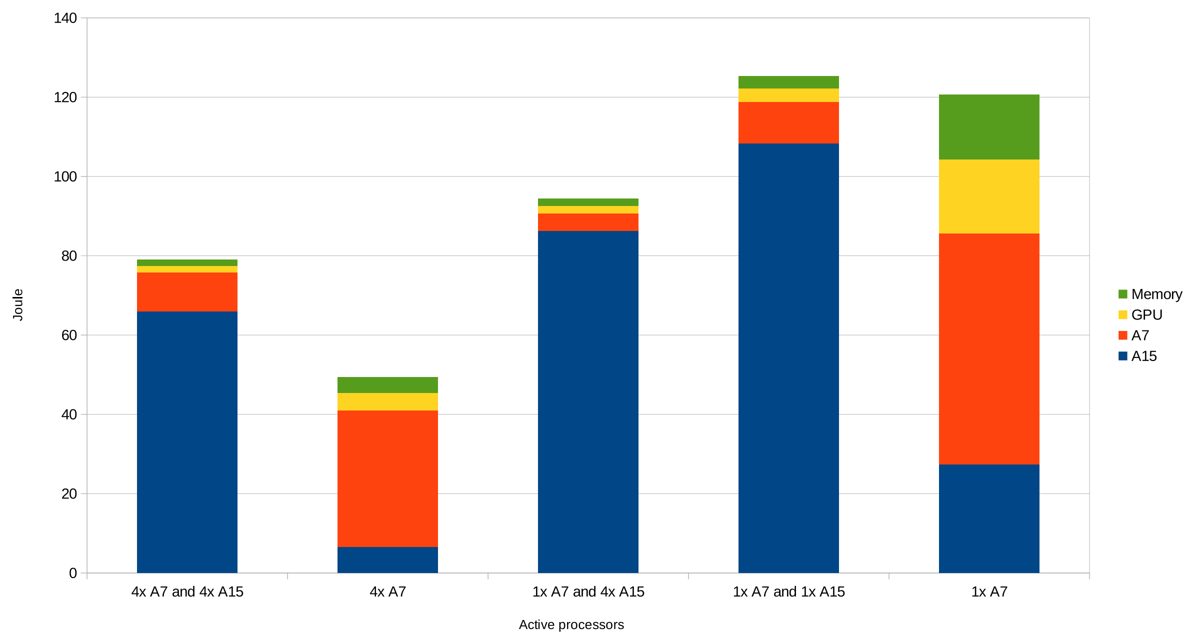

Figure 6.6: Total energy consumption execution the whole 2D-Convolution program. The colors indicate different components, and the full height of each column is the accumulated total power consumed for the whole system.

The data here show that, even though the fully powered system with all eight processors operating performed better than all the other processor configurations, it is not the best at energy consumption. The power consumed by a full execution of 2D-Convolution on all 8 processors consume 60% more power than running it on the 4 Cortex-A7 cores alone. Running on the more energy efficient, processors cost performance, but it save energy. This is a trade off that is often observed in energy efficiency. It vary from application to application what is the preferred property, but it is always an issue of balancing the two. In section 6.5 Energy Delay Product (EDP) measurements the balance point between performance and energy of these experiments are explored in further depth.

Apart from the results for the full eight core system and the 4 small cores, experiments for the other processor configurations were also carried out. These results however, were not promising. All the other processor configurations had higher energy consumption than the two first configurations. They their performance was also worse than the full system, which completed the execution with a lower power consumption. In other words, they did neither stand out in energy efficiency or performance. There may exist potential situations with very specific maximum execution times and such, where these configurations may be viable. They are however not promising candidates for general load distribution strategies.

6.5 Energy Delay Product (EDP) measurements

While the energy consumption is a good energy measure in many situations, it is often interesting to compare the energy consumed with it's performance trade off. As explained in section 3.1 Energy measurement energy delay product measurements can be used to examine this trade off. These are the energy data from chapter 6.4

Chapter 7

Conclution

Chapter 8

Future Work

These are some suggestions for future work that may build upon the work in this thesis.

8.1 Experiment with heterogeneity

In this thesis, there have been done experiments with the Exynos 5, which support ARM big.LITTLE. The heterogeneous properties of this processor was outside of the scope of this pilot project. The same applications can be adapted and optimized to explore the potential of this processor architecture. This is planned for the master thesis following this pilot project.

8.2 OmpSs with OpenCL kernels

A new feature of OmpSs is it's ability to manage OpenCL kernels as tasks. It is possible to issue OpenCL kernels as OmpSs tasks, and have the task manager assign them to GPUs and CPUs. This allow for portable code that can run effectively on a range of different system. It would be interesting to examine the potency of this way of utilizing the GPU, as it save the programmer from the job of manually tuning the load balance between GPU and CPU.

8.3 ARMv8-A 64-bit processors

ARM have created the next generation ARM processors. They run a new instruction set, with support for both 32- and 64-bit instructions. Running similar experiments on such a processor would be interesting.

8.4 Performance measurement

In this project, only simple forms of performance measurement was used. Running the same or similar experiments with access to other performance counters would be interesting. Cache hit rate and utilization, memory utilization, bus utilization and other counters could help understanding the strengths and weaknesses of the system.

Appendix A

Implementation

A.1 Introduction

A.1.1 Program 1

Bibliography

[1] Hardkernel.com. http://www.hardkernel.com/main/products/prdt_info.php?g_code=G140448267127&tab_idx=2. Accessed: 2014-10-25.