ÇANKAYA UNIVERSITY
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

MASTER THESIS

PERFORMANCE COMPARISON OF MULTI-CORE SMARTPHONES

Mustafa Maan ALSABBAGH

MARCH 2018

PERFORMANCE COMPARISON OF MULTI-CORE SMARTPHONES

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED
SCIENCESOF
ÇANKAYA UNIVERSITY

BY
MUSTAFA MAAN ALSABBAGH

IN PARTIAL FULFILLMENT OF THE REQUIREMENTSFOR THE
DEGREE OF
MASTER OF SCIENCE
IN
THE DEPARTMENT OF
COMPUTER ENGINEERING

MARCH 2018

Title of the Thesis: **PERFORMANCE COMPARISON OF MULTI-CORE SMARTPHONES**

Submitted by: **MUSTAFA ALSABBAGH**

Approval of the Graduate School of Natural and Applied Sciences, Çankaya University.

Prof. Dr. Can ÇOĞUN
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

Prof. Dr. Erdoğan DOĞDU
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

Asst. Prof. Dr. Sibel TARIYAN ÖZYER
Supervisor

Examination Date: 16.03.2018

Examining Committee Members:

Assoc. Prof. Reza HASSANPOUR (Çankaya Üniversitesi) _____

Asst. Prof. Sibel TARIYAN ÖZYER (Çankaya Üniversitesi) _____

Asst. Prof. Shadi ALSHEHABI (THK Üniversitesi) _____

STATEMENT OF NON-PLAGIARISM

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name : Mustafa ALSABBAGH

Signature :

Date : 16.03.2018

ABSTRACT

PERFORMANCE COMPARISON OF MULTI-CORE SMARTPHONES

ALSABBAGH, MUSTAFA MAAN
M.Sc., Computer Engineering Department
Supervisor: Asst. Prof. Dr. Sibel TARIYAN ÖZYER
MARCH 2018, 54 pages

Smartphones are an important communication tool, having tendency to emerge as the mobile computer, that is, presenting all functions needed on the way of an individual. Subsequently, multi-core smartphones have been manufactured, therefore applying parallel processing on these devices have become possible. Using parallel processing in smartphones has many advantages, such as reducing processing time, power optimization, use the full power of the smartphone and giving us the ability to write more efficient applications. In this thesis, the way to write parallel applications will be investigated on multicore smartphone, and analyzed the dependency for the applications with the purpose of determining the parts of the code within the application that will be implemented in parallel. The Bernstein's conditions will be applied which describe when two program fragments can be executed in parallel. The way determining the most favorable quantity of threads will be indicated to be used in parallel application. The success in gain of performance will be attained by parallel processing of application on more than one processor, or what is called Speedup. Java 2 Micro Edition (J2ME) will be used as a programming language to develop mobile phone applications and Android Studio as a developing environment.

Keywords: Mobile phone, parallel programming, lane detection, multi-core processor.

ÖZET

ÇOK ÇEKİRDEKLİ AKILLI TELEFONUNLARDA PERFORMANS KARŞILAŞTIRMASI

ALSABBAGH, MUSTAFA MAAN
Yüksek Lisans, Bilgisayar Mühendisliği Anabilim Dalı
Tez Danışmanı: Yrd. Doç. Dr. Sibel TARIYAN ÖZYER
Mart 2018, 54 sayfa

Akıllı telefonlar önemli bir iletişim aracıdır ve bireylerin ihtiyaç duyabileceği tüm özellikleri sunan taşınabilir bir bilgisayar haline gelme eğilimindedirler. Bunun sonucunda, çok çekirdekli akıllı telefonlar üretilmiş ve bu nedenle bu cihazlarda paralel işleme uygulaması mümkün hale gelmiştir. Akıllı telefonlarda paralel işlemenin kullanılmasının işleme süresinin kısaltılması, güç optimizasyonu, akıllı telefonun tam gücünü kullanma ve daha verimli uygulamalar yazabilme yeteneği sağlaması gibi birçok avantajı bulunmaktadır.

Bu tezde, paralel uygulamalar yazmanın yolu çok çekirdekli akıllı telefon üzerinde incelenecek ve paralel olarak uygulanacak uygulama içerisinde kodun bölümlerini belirlemek için uygulamalara olan bağımlılık analiz edecektir. İki program parçasının paralel olarak yürütülebileceği zamanı tanımlayan Bernstein koşulları uygulanacaktır.

Optimum sayıda iş parçacığını belirleyen yol paralel uygulamada kullanılmak üzere gösterilecektir. Uygulamayı çalıştırarak elde edilecek performans kazanımı, Speedup (hızlanma) olarak adlandırılan çoklu işlemciler üzerinde paralel olarak gösterilecektir.Cep telefonu uygulamalarını geliştirmek amacıyla programlama dili olarak Java 2 Micro Edition (J2ME), geliştirme ortamı olarak ise Android Studio kullanılacaktır.

Anahtar Kelimeler: Mobil telefonlar, Paralel pragramlama, Lane belirleme, Çoklu çekirdekli işlemci

ACKNOWLEDGEMENTS

I am grateful to Allah for giving me the good health and ability being that were necessary to complete this thesis.

I would like to thank my supervisor, Asst. Prof. Dr. Sibel TARIYAN ÖZYER, for her guidance and encouragement. she has given scientific and technical support to show the fullest picture of project.

First of all, I am really grateful to my dear father who keeps giving me moments of happiness, throwing away all the obstacles and difficulties, and paving for me the way to success.

Dear mother who always gives me love and compassion. She is the symbol of true love, healing balm for the heart, and pure whiteness.

I would like to thank my sisters especially Safa AL-SABBAGH and Noor AL-SADEY who with their kind and pure hearts supported me much.

Finally, my great thanks to my friends Mohammed YAHYA, Mohammed AL-HAFIDH, Ahmed ABDUL MAJEED and all other friends for what they have done for me.

TABLE OF CONTENTS

TABLE OF FIGURES

CHAPTER 1

INTRODUCTION

Mobility is one of the core terms in the contemporary world. Disregarding your location or activities, the universe of mobile devices is all around you. Desktop terminal workplaces do not confine us anymore, communication and work are now possible from practically any place. The advancement in the miniaturization world has implemented this new way of interaction. Nowadays, we can communicate and work via multiple devices like PDA's, Laptops, mobile phones and Ultra Mobile PC's.

Although these devices might surround us, we must raise the question: are they used to their full potential? The field of parallel processing contains a potential solution to this, since without disclosing the real capacity of a system with embedded multiple cores, this new technology is not fully used.

In our daily computing, systems with multiple cores for personal computers remain a significant area, for multi-core processors vast range of APIs and frameworks that focus on parallel programming have been suggested; these attain maximum utilization of processor and speedup. Though, in the field of systems with multiple cores for smartphones very few studies have been offered fundamentally since this is a comparatively fresh notion and few final products have adopted the system. Recently released smartphones with multi-core, like Samsung Galaxy SII and SIII, enhanced efficiency of power consumption and performance, have transformed mobile computing and rendered new paradigms of study possible, particularly with regard to real time data processing.

In this way, using complex algorithms can be considered for prospective adaptation that formerly was considered to be impractical to implement on smartphones'

platforms. For example, conducting specific calculations on large data matrices on smartphone in the past was very problematic, resulting partly from its memory restrictions, but basically from the processing power of the smartphone. The possibility to use complex algorithms is desirable when possible since these commonly provide more precise results. Recently released multi-core smartphones fortunately have enabled us to do precisely that because it became possible to achieve true parallelism.

A parallel algorithm and the machine that executes it comprise a parallel system, and either components compute with a few variants. Concerning parallel algorithms, it is possible to specify the variables applying a vast variety of paradigms and models. As for supporting architectures, although all of them compute using multiple processors, it is possible to have them different in more than one dimension, like in the interconnection network, the control mechanism, the granularity of processors, and the address space organization. Amid the purposes of parallelism there are those of decreasing the runtime and using resources for computing efficiently [1].

1.1 Parallel Processing Concept

Parallel computing implies the notion of speeding up the processing of a task through splitting it into more than one fragment that are possible to be processed concurrently, every fragment over their own processor. If a program is processed over **N** processors, performance would probably be n times faster than it would be when run just on one processor [2].

1.2 The advantage of using Parallel Processing

When one of the first computers were used, just a single program could run at a time. A program for copying tape taking an hour to execute and an execution comprehensive program taking an hour to execute would totally take two hours to fulfil their function. One of the initial forms of parallel processing empowered simultaneous performance of both programs. An input/output operation would be started by the computer, and during the time it was expecting for the completion of the operation,

this could perform processing time of the processor. The overall time of performance for tasks which are more than one would total to a bit more than an hour [3].

1.3 Parallel Processing in computers

Processors that work in parallel are computer systems that consist of numerous units for processing being linked through a network of interconnection along with the software that is necessary for uniting the work of processing units. There is a couple of main factors of these systems' categorization: the network of interconnection that links the processing units together and the units themselves. The units of processing are able to interact and communicate with each other via means of message passing or shared memory. The network of interconnection for systems of divided memory could be categorized as switch-based against bus-based. In systems of message while pass through, the network of interconnection gets split in to dynamic and static. Dynamic links create connections as the program executes, on the fly. Static links are of steady topology not changing while programs are running. The major advantage of implementing multiprocessors is creating powerful computers through plainly linking numerous processors. A multi-processor is supposed to develop higher speed than a system with the fastest single-processor. Moreover, a multi-processor containing many of every processors is supposed to be much more paid-efficient than producing the only processor having big exhibition. Other argument for a multi-processor is fault resistance. When a processor fails, continued service should be provided by the other processors, even if the performance degrades [4].

1.4 Contribution

In this project, we consider an implementation of lane departure warning systems, having data parallelism and the principles of task as the source, this work proposes an approach of parallel programming on mobile devices with octa-core to recognize lanes on the road and send an alert to the driver in case the vehicle is departs from the lane, which features properties as follows:

1) Implement parallel algorithms on mobile phone with multi-core.
2) Display a way to escalate processor's utilization to improve the running time of the system.
3) Design and develop lane departure warning system that runs on multi core smart phones that uses the full power of the multi core processer.
4) Propose and design detection filter to extract the whole line not only the edge of the line like in the standard edge detection filters.
5) Show the full potential of the multi core smart phones.

CHAPTER 2

PARALLEL PROCESSING

Parallel Processing is a computing type where numerous calculations are performed concurrently, functioning on the basis that big tasks can frequently be chunked into smaller parts, that are later solved concurrently ("in parallel manner"). Parallel Processing can be categorized into several various forms: parallelism of tasks, data, level of instructions, and level of bit. Parallelism has been applied for a long time, mostly for high performance processing; however, its significance has boosted recently for the reason of the physical limits constraining the possibility of frequency measurement. Since power processing (subsequently generating heat) carried out by computers has recently turned to be a problem, computing in parallel manner has evolved into the predominant concept among computer architectures, fundamentally in the representation of processors with multicore [5].

It takes more effort to program software of parallel computer than sequential programs since the simultaneousness yields some new categories of possible bugs in software, having race conditions as the most widespread ones. Simultaneousness and communication of the various sub-tasks commonly become one of the biggest obstruction for achieving proper performance of the parallel program [6].

2.1 Major applications of Parallel Processing:

- Parallel processing is used by scientists in order to create prototype of cars and crash barriers generated by computer to detect the firmness and tolerance of the crash barriers in the case of an accident. Using a machine with a single processor, one prototype test can last up to five days. Using a

parallel computer, the it is possible to divide prototype into parts, every of which travels to their own processor, completing a process that generally lasts five days in a couple of hours.

- Parallel processing is used by airlines for processing information of customers, estimate demand and determine the prices to be paid.
- Supercomputers with parallel processing is used by the medical community for analyzing MRI scans and analyze prototypes of bone implants' systems.
- Parallel supercomputer computing supplies can also be applied for cracking codes of encryption, structural analysis, chemistry, physics, animated graphics, analysis of geological data, meteorology, electronic design, and computational fluid dynamics [1].

2.2 Computer architecture taxonomy by Flynn

The most common of computer architecture taxonomy was proposed by Flynn in 1966. The notion of an information stream lies in the base of this scheme of classification. Two kinds of information stream to the processor: data and instructions. The data streaming can be categorized as exchange of data traffic between the processing unit and the memory. The instruction stream can be categorized as the performance of chain of instructions by the processing unit. In accordance with classification by Flynn, either of the data or instruction streams, are able to be multiple or single. The architecture of computer can be divided into 4 specific sections as follows [7]:

- Single instruction single data streams (SISD);
- Single instruction multiple data streams (SIMD);
- Multiple instruction single data streams (MISD); and
- Multiple instruction multiple data streams (MIMD).

Common von Neumann computers with single processor fall into the category of SISD systems. Computers processing in parallel can either be MIMD or SIMD. The parallel machine is categorized as SIMD when having single control unit and all the

processors perform identical instruction synchronously. In machines of MIMD type, every processor is capable of operating on varying instructions on various data and has a control unit of its own. The category of MISD has identical data stream flow into a processors' linear array that perform various streaming of instruction. Practically, no viable MISD machine exists; though, few authors have classified pipelined machines, and probably computers with systolic array, to be MISD examples [8]. Figures 1, 2, and 3 illustrate the block diagrams of SISD, SIMD, and MIMD, accordingly.

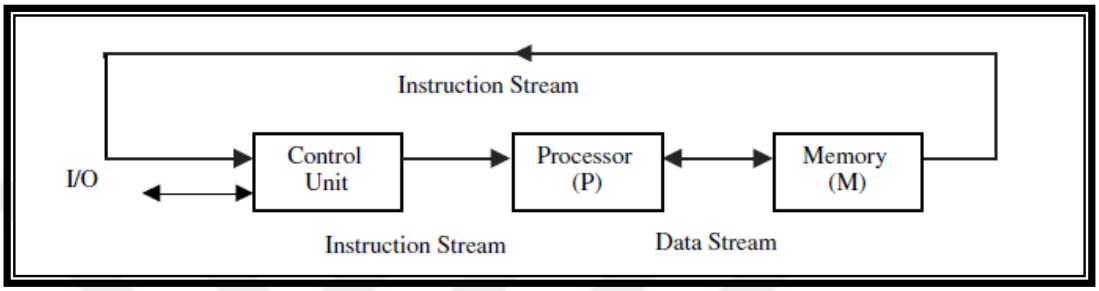

Figure 1. SISD Architecture.

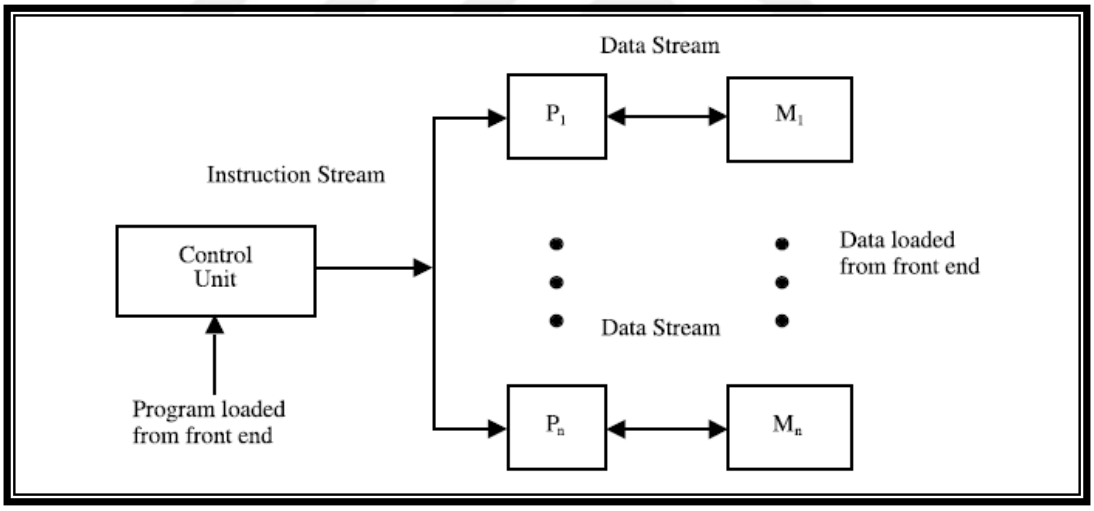

Figure 2. SIMD Architecture.

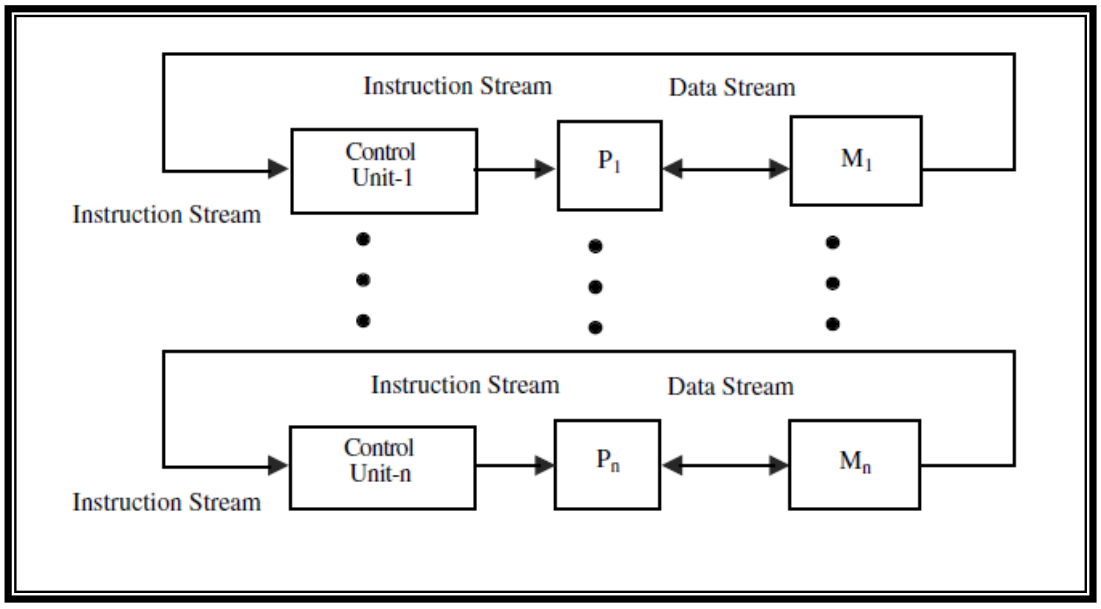

Figure 3. MIMD Architecture.

2.2.1 Architecture of SIMD

The SIMD prototype of parallel programming is composed of 2 features: a front end computer of the conventional von Neumann type, and an array of processors depicted by Figure (4). The array of processor is a batch of processing components that are identically synchronized and can simultaneously execute the same task using varying data. Every processor within the array possesses a little portion of local memory hosting the distributed data while its parallel computing is performed. The array of processor gets linked to the front-end memory bus allowing front-end to randomly access the memories of local processor taking it as another memory. Accordingly, the front-end can distribute specific instructions causing memory parts to be executed concurrently or trigger data to circulate in the memory. It is possible to design and perform a program on the front end on the basis of a traditional serial language of programming. The program of application is performed by the front end implementing normal sequential method, but distributes commands to the array of processor for SIMD processes to be carried out in parallel [9].

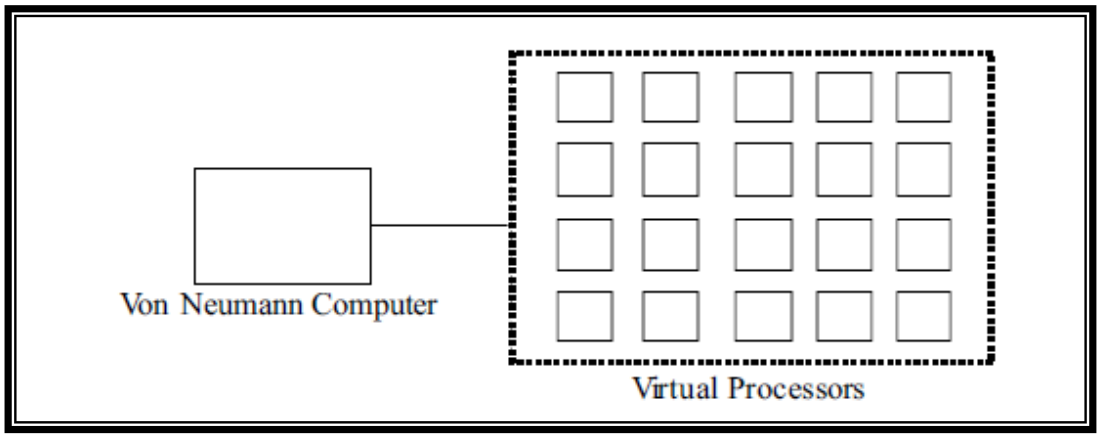

Figure 4. SIMD Architecture model.

The correspondence between serial and data parallel computing is considered to be one of the strongest aspects of data parallelism. Lockstep type of processors' synchronization turned synchronization to irrelevancy. The processors either perform no process or identical simultaneous operations. In architecture of SIMD, parallelism is used by implementing concurrent operations between big data sets. This notion yields most use when solving problems having plenty of data to update on a wholesale basis. It is specifically effective among plenty of standard numeric calculations. The machines of SIMD type have used two primary configurations (see Figure 5). In the first pattern, every processor features a local memory of its own. The interconnection network is used to provide communication across processors. When straight link between a specific couple of processors is not provided by the interconnection network, the pair can use an intermediate processor for exchanging data. Such interconnection pattern was used in the ILLIAC IV. The interconnection scheme in the ILLIAC IV availed direct communication of each processor with four neighboring processors in a pattern of 8*8 matrix to ensure direct communication of the i^{th} processor with the $(i-1)^{th}$, $(i+1)^{th}$, $(i-8)^{th}$, and $(i+8)^{th}$ processors. In the second pattern of SIMD, the interconnection network is used by memory modules and processors to communicate with each other. Two processors are able to exchange data through module(s) of intermediate memory or perhaps through intermediate processor(s). The second SIMD pattern was used by the BSP (Burroughs' Scientific Processor) [10].

2.2.2 Architecture of MIMD

Parallel architectures of multiple instruction multiple data streams (MIMD) consist of multiple memory modules and multiple processors that use some interconnection network to connect together. They are split in to two vast categories: message passing or shared memory. Figure (6) depicts the overall architecture of the two categories [11].

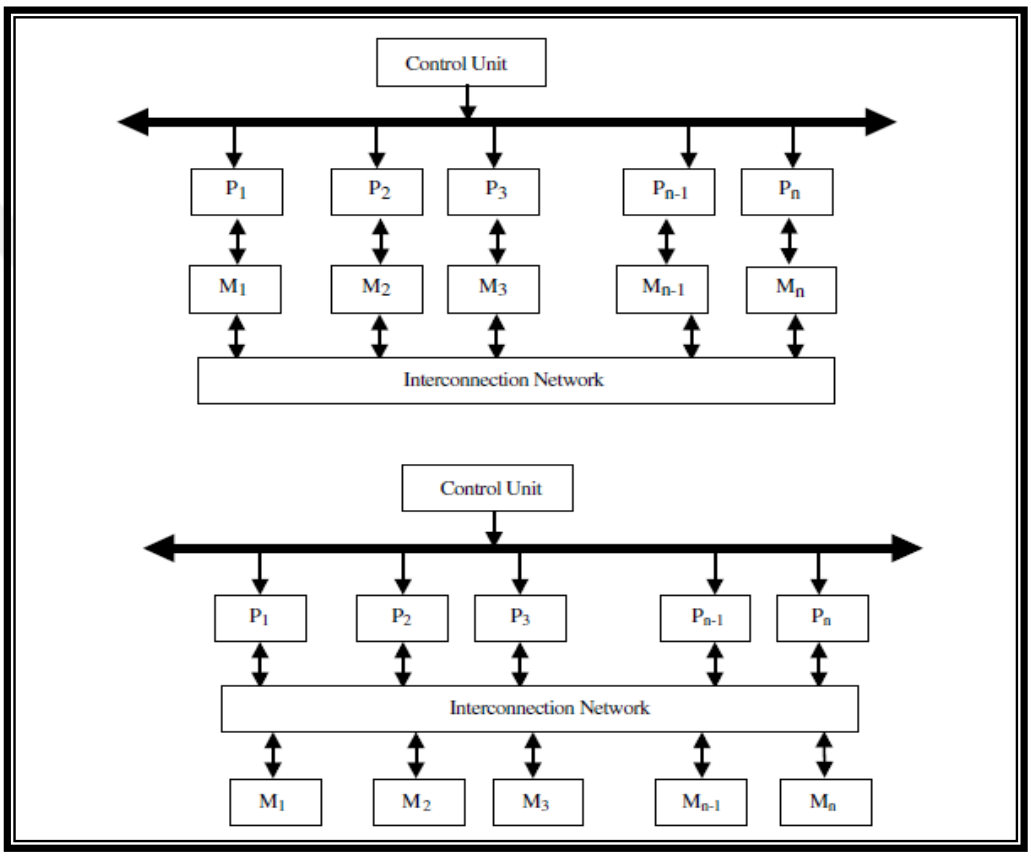

Figure 5. Two SIMD schemes.

Processors transfer information between each other via their central joint memory in systems of split memory, and transfer information between each other via their network of interconnections in systems of passing messages.

The inter processor coordination of a joint memory architecture is generally acquired via global memory that all processors share. Those commonly being server systems communicating using a controller of bus and cache memory. The architecture of cache/bus decreases the necessity of costly multiport memories and circuitry of interface along with the necessity of adoption of a paradigm of message passing while

developing software for application. Due to balanced access to shared memory, the systems are as well named systems of SMP (symmetric multi-processor). Every processor possesses an identical opportunity for reading/writing to memory, as well as identical speed of access [12].

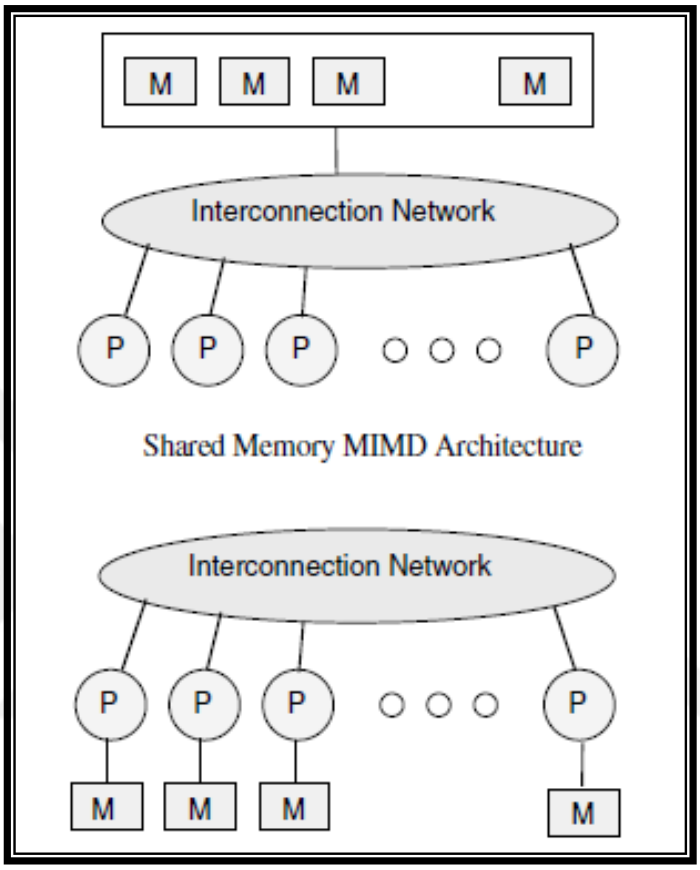

Figure 6. Shared memory versus message passing architecture.

2.3 Dependencies

The fundamental process in application of parallel algorithms is understanding data dependencies. There is no program that is able to perform faster than the longest dependent calculations chain, what is referred to as the critical path, as the calculations dependent upon previous calculations within the sequel are to be processed consecutively. Nevertheless, the majority of algorithms are not composed of just a long dependent calculations chain. The opportunities to perform independent calculations in parallel are usually available [13].

Let two program fragments be P_i and P_j. Bernstein's conditions describe when it is possible to execute the two in parallel and when they are independent. For the P_i variable, let us choose I_i as the input variants and O_i as the output variants, and accordingly for P_j. P_i and P_j are not dependent variables in case the variable meets:

- $Ij \cap Oi = \Phi$
- $Ii \cap Oj = \Phi$
- $Oi \cap Oj = \Phi$

When the first condition is violated, it presents a dependency on flow, conforming to the first statement introducing a result used by the second statement. An anti-dependency is represented by the second condition anti-dependency, where a variant necessary for the initial expression (P_i) would be overwritten by the second statement (P_j). An output dependency is represented by the third and last condition: if the two statements register to a sole location, the last result must be from the consecutive statement that was performed last.

The functions as follows due to be considered: those present some types of dependencies:

1: function Dep (a, b)

2: c := a·b

3: d := 2·c

4: end function

The task 3 in Dep (a, b) is impossible to be processed prior to (even simultaneously with) the task 2, as task 3 applies an outcome of task 2. The first condition becomes violated, and as a result creates a flow dependency.

1: function NoDep (a, b)

2: c := a·b

3: d := 2·b

4: e := a+b

5: end function

There are no dependencies among the commands in this instance, thus all of those can be executed in parallel. The conditions of Bernstein prohibit memory from being split across varying operations. Therefore, several ways of carrying out an access ordering is needed, like barriers, semaphores or several other means of synchronization [35].

2.4 Hardware and Software For the implementation of parallelism, we need special hardware and software support. In this section, we are going to address these support issues. We first distinguish between hardware and software parallelism. We describe the basic concept of compilation support that is necessary for closing up the crack between software and hardware.

The key idea being conveyed is that parallelism cannot be achieved free. Besides theoretical conditioning, joining efforts between hardware designer and software programmers are needed to exploit parallelism in upgrading computer performance**Parallelism**

.

2.4.1 Hardware Parallelism

Hardware parallelism represents the kind of parallelism determined by the structure of machine and multiplicity of hardware. The paradigm is often a task of performance and cost compromises. It presents the patterns of reserves utilization of operations executable concurrently. It is also capable of indicating maximum power of execution of the processor resources.

Parallelism in a processor can be characterized by the quantity of commands issued during one cycle of machine. If a processor sends **k** commands during one cycle, the processor is then classified as a **k-issue**.

A conventional processor takes one or more cycles of machine for issuing a sole command. Such kinds of processor are named **one-issue** machine, working on a

sole pipeline of instructions in the processor. In a present-day processor, there can be a couple or more commands issued during one cycle of machine.

A multiprocessor system built with **n** processors of **k-issue** type are expected to process a peak number of **nk** threads of instructions concurrently.

2.4.2 Software Parallelism

This type of parallelism is identified by the data and control dependence of the programs. The parallelism scale is disclosed in the flow-graph or profile of the program. Software parallelism is the algorithm's functions, optimization of compiler, and the style of programming. The flow-graph of a program depicts the pattern of operations that can be executed concurrently. In the course of the operation span parallelism in a program changes. It often confines the maintained work of the processor [3, 7].

2.5 Speedup

The extent of performance gain attained by executing our operation in parallel on multiple processors is explained by the speedup of code. The simplest way to define it: the period of time consumed by a program to be performed on a single processor, divided by the time consumed by it to be performed on multiple processors. The range of Speedup is most commonly identified between 0 and **p**, where **p** means quantity of processors. The following formula is used to define Speedup:

$$S_p = T_1 / T_p \quad \ldots\ldots\ldots\ldots\ldots\ldots \quad (1)$$

With:

- **p** – quantity of processors;
- T_1 – time spent to execute algorithm in sequence;
- T_p – time spent to execute algorithm in parallel on **p** processors;
- When **Sp=p, ideal speedup** or **linear speedup** is attained**.** If an algorithm is run using linear speedup, doubled quantity of processors multiplies the speed by two. Considering this perfection, the scalability becomes really high.

2.5.1 Linear Speedup

If it takes a processor a quantity of time **t** to execute a process and if **p** number of processors are able to execute a process in time **t/p**, then linear or perfect speedup (**Sp=p**) will be attained.

- To put it another way, operation on four processors advances the time by a factor of four, operation on eight processors advances the time by a factor of eight, etc. [14].

2.5.2 Speedup Extreme

The speedup reaches its peak if it is:

- larger than p, named super linear speedup;
- smaller than one. [14]

2.5.3 Super Linear Speedup

In parallel computing we sometimes can observe a speedup of higher than **p** while operating on **p** processors, that is known as a super linear speedup. Such a kind of speedup seldom occurs and frequently gets starters confused, believing that the peak speedup in theory should reach **p** when using **p** processors.

A probable background of a possibility of super linear speedup application lies behind the effect of cache that result from the varying memory scales of a conventional computer. In parallel programming, the quantity of processors varies, but also does the volume of accrued cache storages from various processors. Possessing bigger size of accrued cache, greater amount or probably full work set is able to house cache storages and the time spent on accessing memory decreases significantly, causing the appearance of additional speedup extra to the speedup from the computation itself.

2.6 Efficiency

Parallel performance is measured by efficiency which holds close relation to speedup and is frequently also included into parallel program performance description. It is a scale, generally from 0 to 1, that estimates the quality of processors utilization when those solve a problem, in comparison to the amount of an effort spent on synchronization and communication. The efficiency of the systems using linear speedup and systems operating on one processor equals to one, when the efficiency of vast quantity of systems that are difficult to run in parallel is equal to 1/ln(**p**) having the result around zero according to the quantity of increased processors.

- Efficiency with **p** processors gets determined as the proportion of speed-up using **p** processors to **p**.

  Ep= Sp/p …… (2)

- Efficiency is a decimal generally ranging from zero to one.

  Ep=1 equals to an ideal speed-up of **Sp=p**.

The efficiency can be described as the median speed-up for a processor. [15] [16].

2.7 The Necessity of Multiprocessing

A mobile device can execute a vast range of processes like SMS text messaging, video playback, services based on location, mobile gaming, and browsing Web. Considering the expanding supply of Wi-Fi and high-speed mobile networks, different processes requiring high performance can be executed by mobile devices that before were operated on conventional computers. The upcoming series of smart phones also known as “Super phones” and tablet computers are to handle a vast range of processes like video editing, HD 1080p videos playback, running on-line games based on Adobe®/Flash®, operating visually rich gaming, streaming HD videos based on Flash, downloading high-definition videos concurrently, encoding and uploading, and conferencing with high-definition real time video calls.

The present-day series of processors in mobile devices were not developed to handle such an intense line of cases that require performance to be high. The users’

experience quality while using a device with a single-core CPUs significantly decreases if a few apps are executed by a user simultaneously, or apps that require high performance are processed like video editing, video conference, games, and so on. Some techniques are employed by engineers addressed at the improvement of CPU performance. Some of those are: using bigger on-die cache storages, and using bigger cores, boosting voltage and frequency of core processing, applying smaller and faster semiconductor tasks.

Expanding the cache or CPU core's volume boosts the performance just to a specific point, by overpassing the point issues concerning heat and thermal dissipation turn any following the expansion of cache's and core's sizes not practical. Fundamental physics of semiconductors states that expanding processing voltage and frequency can prominently boost power consumed by the devices of semiconductor type. Although there is a possibility that engineers might reach high performance if voltage and frequency increases, the life of battery would be strikingly reduced. Moreover, high-power processors would need bigger solution to cool that would result in an untoward increasing of devices' size. Thus, expanded frequency of processing to satisfy the demand for permanently increasing operation of mobile apps is an invalid application for the long haul.

To meet the promptly increasing requirements for operation and produce a sleek mobile device, the producers have begun adopting modernized technology units like Heterogeneous Multi-core computing and Symmetrical Multiprocessing. NVIDIA Tegra is considered to be the most improved processor for mobiles in the world designed from scratch as a heterogeneous multiple core SoC ("System On a Chip") system with 2 cores of ARM Cortex A9 CPU type and some other cores built for the purpose of handling functional processes like graphics, video, and audio. A purpose-built core requires less transistors, can process at low frequency, delivers high-performance, and require much less energy than a general-purpose operating core for processing graphics, video, and audio. [17] [18].

2.8 Symmetrical Multiprocessing (SMP)

The technology of Symmetrical Multiprocessing allows mobile processors not merely to gain effectiveness, but as well satisfy top effectiveness requirements and keeping within mobile power budget. An SMP system with multiple cores is represented by the features as follows:

- The architecture contains two or more identic CPU cores.
- All the cores are administered by a single Operating system and have a common shared system memory.
- Every CPU is able to operate independently on various loads of work and when probable, is as well able to share loads of work with the other CPU.

Think of a mobile phone having a CPU with dual core and SMP support; if its application for navigation is executed simultaneously with an audio streaming application, the operating system can distribute the navigation process to one CPU and the audio stream process to the other CPU core. Furthermore, there is also a single multi-threaded application that could use numerous CPUs to its benefit. It is possible that the operating system distributes the threads to execute on both CPUs simultaneously and splits the workload between the two CPUs to complete the process faster. Considering that the workload is joint between the two cores, the cores can process at a lower speed and still achieve outstanding performance as well as conserve power (the voltage required decreases due to processing at lower frequency, that results in a power reduction by the square of voltage decline).

In presented below Figure 1, there is a webpage of common format that is broadly used on the Internet. Various ActiveX and JavaScript-based menu options are contained in it, along with Flash animation and two enclosed Flash videos. A sole core CPU would have to process the whole load of running the content of ActiveX/JavaScript, running Flash player, and the functions of inserted video decoding. Moreover, some background tasks like Twitter® streams, navigation, voice applications etc., will have to be handled by the processor. Performing such complex multitasking functions, the processor commonly runs at peak voltage and frequency, consuming vast quantity of power accordingly.

Using a mobile processor having a CPU with dual core, the function of processing the individual components of the web-page can be split across the two cores of processing. For instance, a core would run Flash video content and background system tasks while the second core would perform Flash Animation, video and ActiveX content.

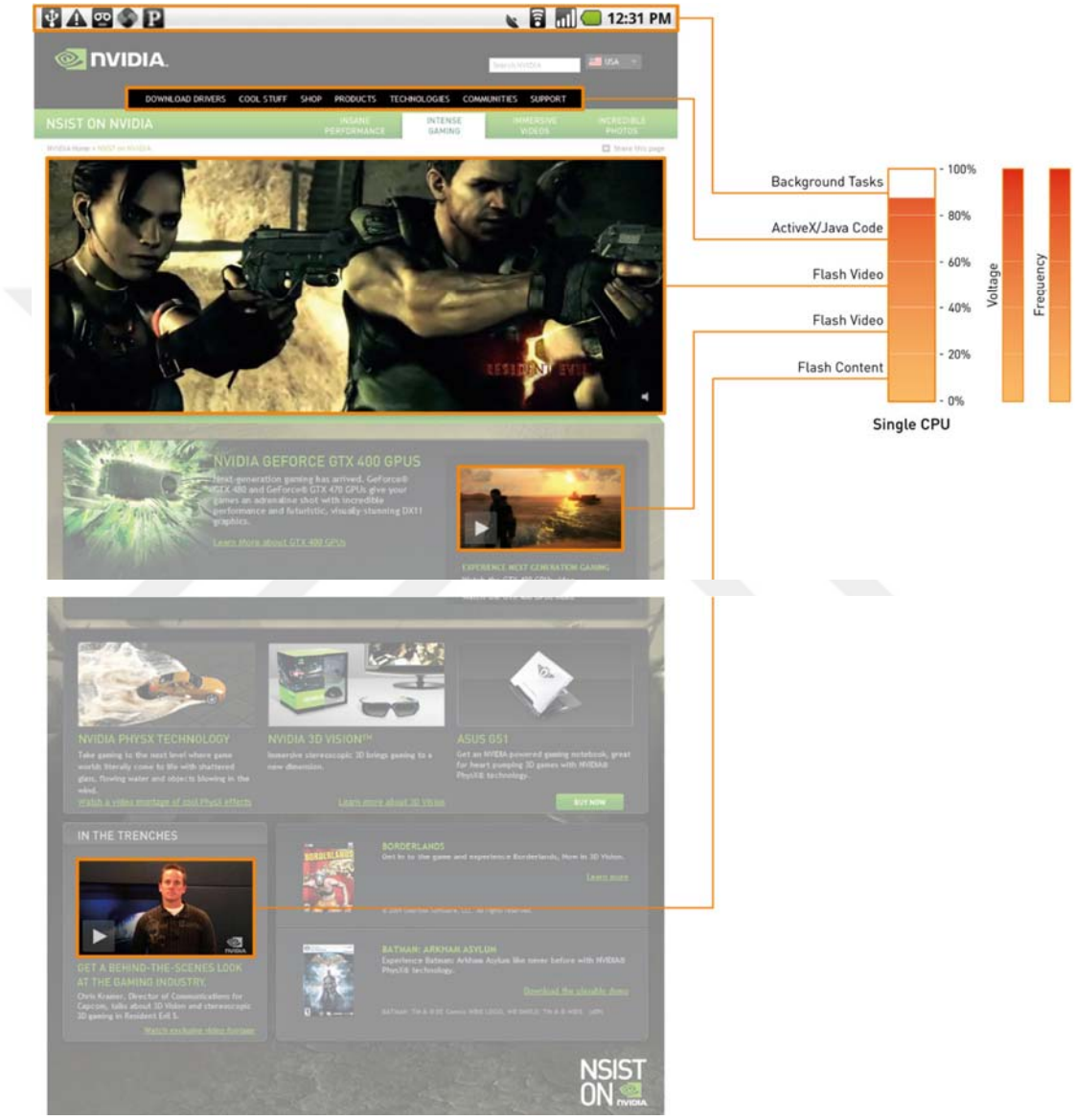

Figure 7. Web page illustrated by a phone with single core.

The below Figure 8 depicts the way of sharing numerous processing tasks between the two cores. As a result of task splitting, the cores are not burdened to perform at their full power and can process at a lower voltage and frequency. Considering the proportionality of power consumption by semiconductor devices to the frequency and voltage-squared, even little operating frequency and voltage

reduction will lead to substantial power consumption reduction. Accordingly, a mobile processor having a dual core CPU and capabilities of SMP will generally yield higher power efficiency than a mobile processor with a single core CPU.

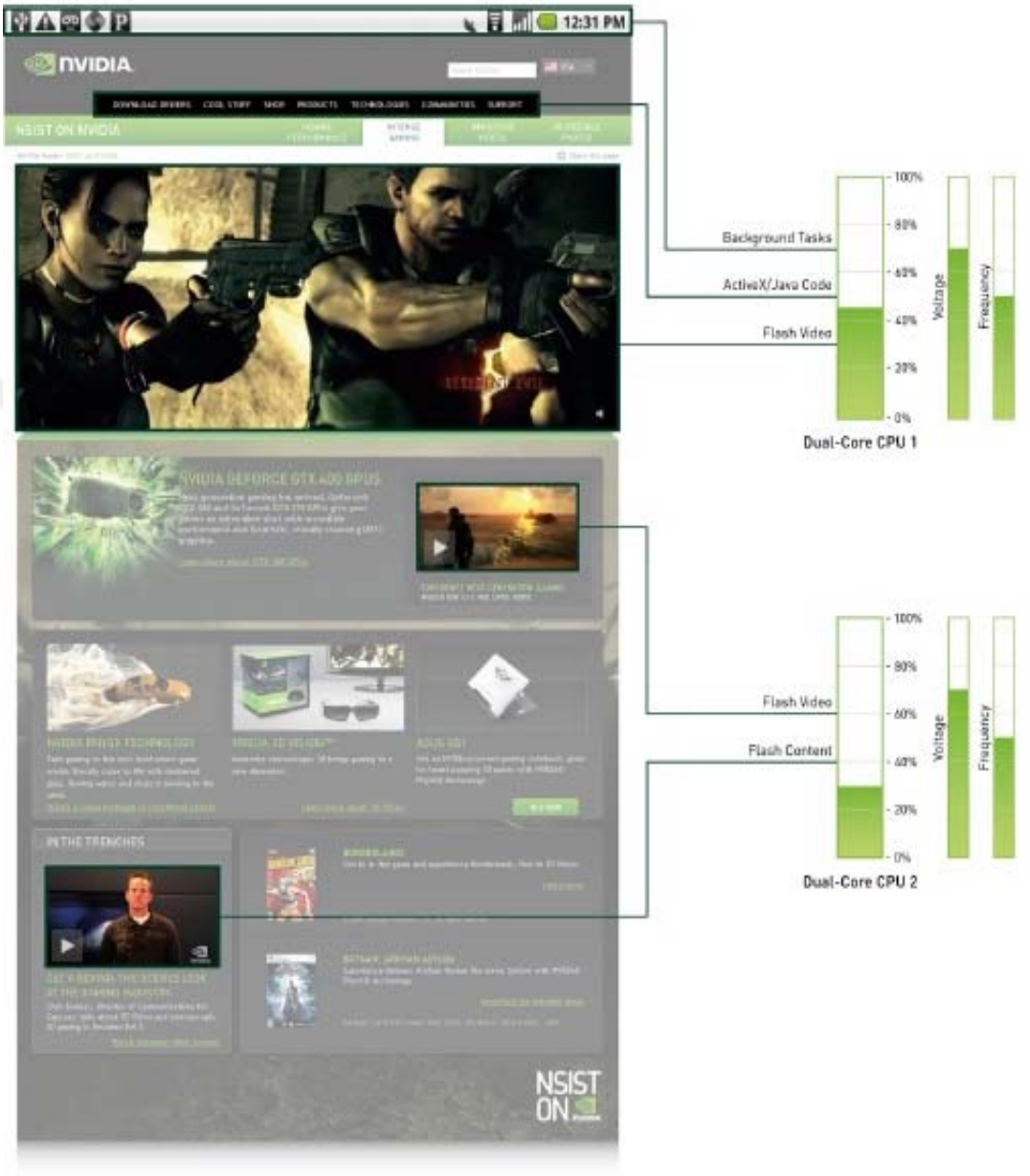

Figure 8. Web page illustrated by a phone with dual core.

CHAPTER 3

LITERATURE REVIEW

There are many researches and previous studies in the area of parallel computing on the computer, however the number of researches in the area of parallel computing in the smart phones are few, and the reason is the smartphones that operate multi-core processor are a modern invention and relatively few.

In 2006, the two researchers explained (Daniel Doolan and Laurence Yang) how can they use mobile message passing interface (MMPI) to perform calculations in parallel across the network of Bluetooth between a set of smartphones, and the researches explained the library component of MMPI and in parallel [19].

In 2009, The researcher (Rudra Hota) and others had presented a system to distinguish street lines in the difficult condition, such as non-clarity lines or Traffic congestion. At first, they divided the picture into two parts: a distant part and nearby part, to be processed each part on its own. Then they performed processing to improve the image, then applied a canny filter to extraction the lines, then the (probabilistic Hough Transform) was applied on the resulting image instead of (Hough Transform), To increase processing speed. The system was tested on a set of videos and the result were good [20].

In 2010, the researcher (A. LÓPEZ) and others had developed a method for extracting the streets lines from the videos depending on the properties in the image, rather than depending on the sharp edges of the lines to extract them, they relied on the distinction of the area located in middle of the line, the image was divided into two parts: right part and left part, assuming that the right part will contain the line in

the right of the vehicle and the left part will contain the line in the left of the vehicle, and then each part will be processed based on that [21].

In the same year, both the researchers and programmers had performed many attempts to design “high level framework” which is provide a layer for programmers, which enables the software engineers to concentrate on writing the application and do not worry about parallel programming. Where developers proposed the framework in the labs of Stanford University in the United States, it consists of two layers: the first is a productivity layer where the developers who work on it have information or little experience in the field of parallel programming, and their focus is on how to write applications. The second layer (efficiency layer) its workers are computer scientists who have experience and extensive background in the field of parallel programming, they have to concentrate on improving the performance and effectiveness of applications [22].

In the same year, the researcher (Panagiotis Tsogkas) suggested in master thesis at the university of Edinburgh skeletons algorithm, the purpose of this algorithm is to take an advantage of multi-core processes by reducing the connection between threads management and regular programming [23].

In 2012, the researcher (Emmanouil Koukoumidis) developed an application running on a smartphone using its abilities to provide many services to the driver, one of them was the expectation signal traffic schedule and when would the signal be green, and the result reduced the fuel rate (20.3%) [24].

In 2012, the researchers (Panya Chanawangsa) and (Chang Wen Chen) had demonstrated the method of proper use of a mobile processor with dual core able to reach immense speed up in mobile applications, and suggest a mobile lane detection system in real time following on from a framework of parallel computing that was correctly designed [25].

In 2013, the researcher (Ding-Yun Chen) and others had proposed a multi core video decoder system level analytics engine (MVSE) to assess the improvement of performance of a platform with a sole core to a platform with dual core applying varying parallel systems. A standard video decoder flow is run by the MVSE on

platform of destination with multiple cores for overcoming the assessment problems of dissonance at accessing memory and inter-communication of cache across various cores, and from testing result display the highest performance was boosted by the hardware of VLD type, reaching ratio of the speedup 2.3 times for a platform with dual core and 3.9 times for a platform with quad core [26].

In the same year, the researcher (Kangwoon Lee) and others introduced a parallel computing technology applying GPU and CPU for the mobile apps for alternate reality. The majority of AR apps are loaded with performing algorithms for comprehensive processing of images to identify specific objects that have cyber interface laid over those. The suggested parallel technology appoints the extraction and description of a component to GPU and CPU accordingly, and those tasks are executed in parallel [27].

In 2014, the researcher (Zongwei Zhu) and others had proposed an innovative framework of memory handling, named Memory Management Based on Thread Behaviors (MMBTB), designed to function on a smartphone system with multiple cores. The framework adjusts to different behaviors of the execution thread via directed optimizations to allocate memory efficiently. The productivity and performance of the innovational memory handling technology for systems with multiple cores was confirmed by an academic simulation prototype, practical testing on actual Android platform demonstrate that MMBTB is able to boost productivity of memory distribution by 12% - 20%, proving academic outcomes of the analysis [28].

In the same year, the researcher (Kuei-Chung Chang) and others had designs of a tool for profiling to study memory behavior of an Android app on the smartphone with multiple cores. The app effectiveness can be improved by designers in accordance with the results of profiling, from empirical results it can be seen that the offered tool is able to study memory condition of the profiled app [29].

In the same year, the (Po-Hsien Tseng) and others had posited mobile apps to be evaluated wrong. Hence, they exploited the app sensitivity notion and invent a scheduler and governor that are user oriented and distribute calculating supplies to apps in accordance with the sensitivity of those. Moreover, they had integrated their

invention into Android OS. The outcomes of comprehensive studies on a mercenary smartphone having real world mobile applications illustrate that the suggested design is capable of achieving considerable achievements in power productivity and also improve user experience quality [30].

In 2015, the researcher (Qianao Ju) and others had evaluated high performance heterogeneous computing on mobile multi-core devices by benchmarking Render Script. Considering the characteristics of the mobile applications, the benchmarks are carefully chosen to explore different behavior in linear algebra, machine learning and image processing. The execution time and power consumption are comprehensively evaluated on the selected multi-core parallel programming on mobile smart phones [31].

In the same year, the researcher (Wonik Seo) and others characterized a mobile platform having a processor with asymmetric multiple cores, researching availability of its parallelism on thread level (TLP) and the influence of cores' asymmetry on apps. This study introduces the effectiveness and power features of a commercial system with asymmetric multiple cores having two types of core. The observation of large and small cores displays the further advantages of using an asymmetric multiple core to improve power productivity. Furthermore, the study researches the availability of parallelism on thread level and behavior of core application of mobile interactive apps via usage of the widespread mobile apps for the Android architecture [32].

In the same year, the researcher (Alejandro Acosta) and (Francisco Almeida) presented a parallel application of the Particle Filter Algorithm on mobile devices with Android and devised 3 different versions of the technology. A sequential application of Java and 2 parallel variants of Render script, an application generated by Paralldroid and an application of special purpose, the outcomes acquired by the parallel variants depict a substantial speedup and high accuracy with a high execution rate of frame per second [33].

In 2016, they showed that a convenient medical ultrasound tool is possible to be composed by implementing a technology with multiple cores and a programmable logic, that combine decreased energy use with great elasticity. An ordinary technology of ultrasound image restoration is discussed along with the possible way of its parallel execution via use of a pipelined design which effectively allocates the tasks across the

features of heterogeneous computing. The design was assessed on the platform of Adapteva Parallella containing an energy productive coprocessor Epiphany with 16 cores and a Zynq SoC containing programmable logic and a processor ARM A9 with dual core. Empirical outcomes depict that parallel beam forming of 128 input channels to a 288x128 pixel ultra-sound image is possible to be acquired on the Parallella with a ratio of 5.3 frames per second using just 2 watts of dynamic power [34].

As we saw in the studies and papers mentioned above, many researchers worked on parallel processing, image processing, and smartphone applications. But very little researchers combined those three field. In our work, we used parallel processing to do image processing using multi-core smartphones. There is one research published in 2012 (Panya Chanawangsa and Chang Wen Chen, 2012" A New Smartphone Lane Detection System: Realizing True Potential of Multi-Core Mobile Devices" IEEE.) he used parallel processing to do lane detection using dual-core smartphone, the differences between our work and this research are that we did lane detection then lane departure waring system using quad-core and octa-core smartphones.

CHAPTER 4

PROPOSED MOBILE LANE DETECTION SYSTEM

The safety of transportation has been a problem of ever rising concern. In 2009, just across the U.S., traffic accidents caused suffering of over 30,000 fatalities, in accordance with the National Highway Traffic Safety Administration. Major quantity of car crashes resulted from the drivers' failure to hold their cars within the allocated lane. As a response to these tragedies, Intelligent Vehicle Assistant System has been attempted to be incorporated into vehicles through adoption of multiple approaches of computer vision.

Although Lane departure warning system has been set up on numerous trucks and other commercial automobiles across North America and Europe and has demonstrated to considerably lessen accidents that are possible to prevent, it still frequently remains optional even for the premium passenger cars. It is frequently quoted that the cost is one of the main issues obstructing a widespread installation of Lane departure warning system. At the same time, it is also important to consider that the installation of such a system, calibration of the camera, and its integration into electronics of vehicle requires professional expertise.

In order to bring Lane departure warning system to the mainstream market, we propose Mobile lane departure warning system, which would assist everyone in using the system. The images of the road are acquired by the system and the lane marks are intelligently labelled. Thus, it determines the direction of the car and its position as related to the boundaries of a road. When the system concludes that the car is close to departing from the ongoing lane, it sends a signal to warn the driver. Lane detection system also includes other functions, such as cruise control autonomous driving and robot navigation. As shown below:

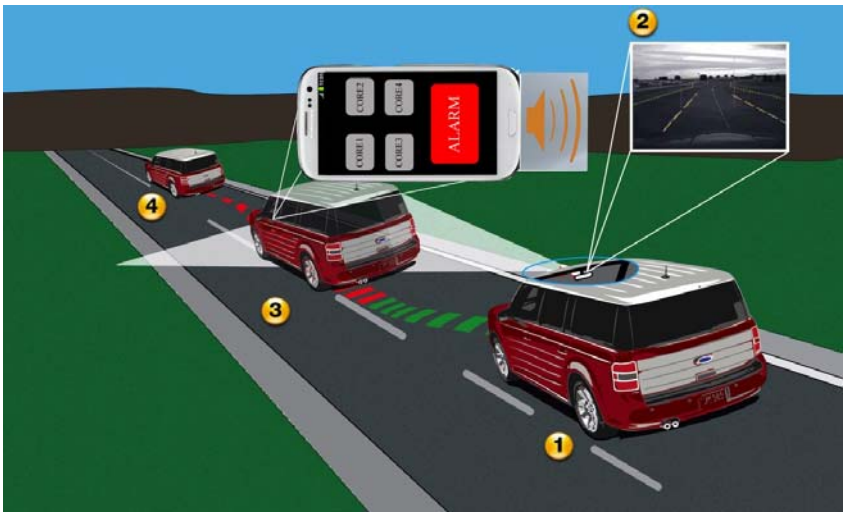

Figure 9. System architecture.

4.1 Development Platform

Android platform was used to develop the system, that is running on multi-core processor smart phone.

4.1.1 Data Acquisition

The phone camera is used for the suggested lane detection system as the source of perception. The raw image frames are captured at the ratio of 30 frames per second (fps) and those are fed to the module of preprocessing.

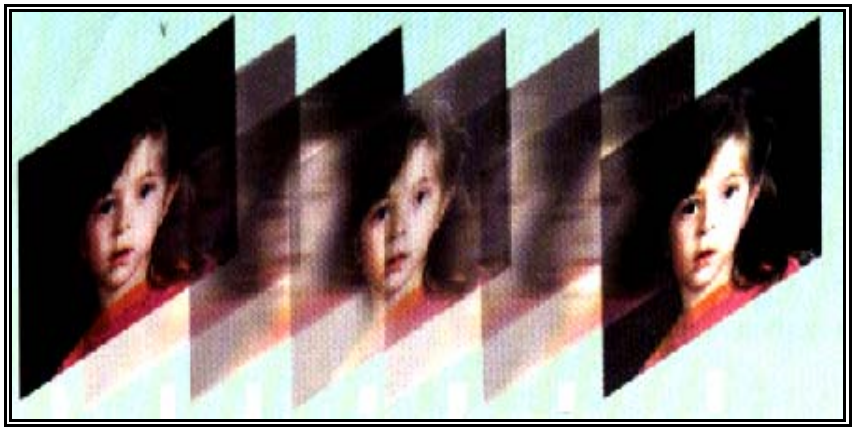

Figure 10. Sequential Video Series.

The phone gets placed onto the car wind-screen by an standard GPS mount. The attachment should be performed in the way that the road is clearly captured by the camera while taking a precaution during this step in the way that the phone does not block the view of driver (Figure 11).

Figure 11. The M-LDWS on a Samsung Galaxy SIII functioning on a standard GPS mount.

API of Android camera avails direct processing of the preview frames by programmers. The storage constraint is possible to overcome due to this feature, since those frames can be processed and the recognized lane boundaries can be illustrated on the phone display leaving the necessity to record any of those. The delivery of frame data is made in byte sets having the default format of YCbCr, for getting the three base colors: red, green, blue.

Figure 12. One of the frames received from the camera.

The frame format is then converted from YcbCr to ARGB8888. Subsequently, an image matrix stores each pixel values. Finally, the basic matrix and linear algebra operations can be implemented.

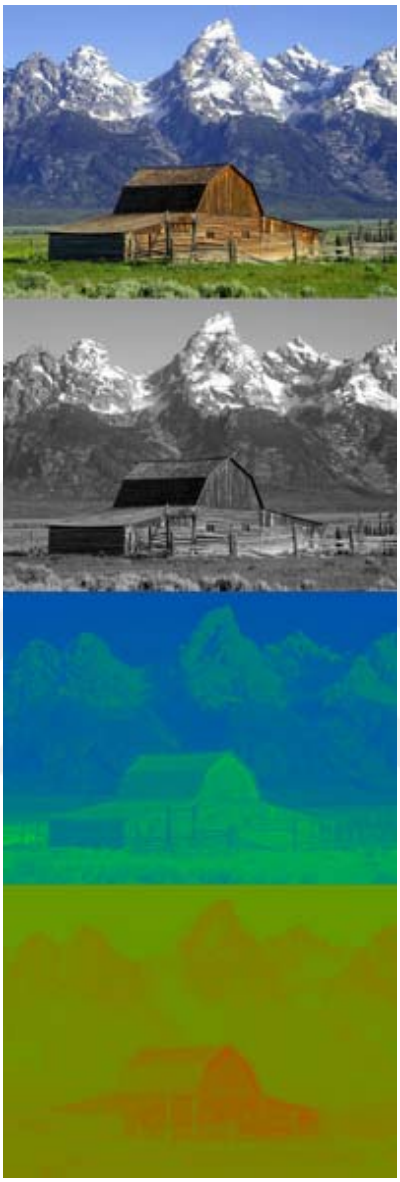

Figure 13. Image format YUV then the same image but using the Y value only then using U value only then using V value only.

4.1.2 Lane Detection

The algorithm of lane identification is established on the presumptions that the definition of the lanes is made by a properly visible white line. For detection of a line, it is important to first identify its most important characteristics: shape and color.

Initially used characteristic for line detection was color. As the road lane's color is white, we execute a procedure of extracting white colour and turning it to green, according to the stated below:

g(x,y) = 255, if r(x,y) >val, g(x,y) >val, b(x,y) >val

g(x,y) = 0, otherwise

Where:

r(x,y) is the red value of the pixel in the position (x,y).

g(x,y) is the green value of the pixel in the position (x,y).

b(x,y) is the blue value of the pixel in the position (x,y).

val is a predefined value.

In the following stage, the feature of shape will be applied to identify the lines, after doing many experiments on a big number of pictures, we designed (5*5) filter which is used to show the whole line not only the edge of the line as in the standard edge detection filters.as shown below:

Table 1. The filter applied to the image.

-1	2	2	2	-1
-1	2	2	2	-1
-1	2	2	2	-1
-1	2	2	2	-1
-1	2	2	2	-1

And to apply the proposed filter on the image, we built a function to do the process of convolution between the filter and the image, according to the equation below:

$$f[x,y] = h[x,y] * g[x,y] = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} h[i,j]g[x-i,y-j]$$

Where:

f: the result image

h: the filter

g: the initial image

m: length of the image

n: width of the image

4.1.3 Lane Departure Warning

After identifying lane boundaries, it is important to detect if the car is within the lane, or it is close to departing from it.

Figure 14. The area is scanned for any lines.

In case the system concludes the car is close to departing from the ongoing lane, it sends a signal to warn the driver. It can be performed by searching a definite section in the picture for any detected lines. In case the detection process result is true, it implies that the car is leaving from the ongoing lane, triggering the system to warn the driver.

Proposed system sequential: After completing the previous steps, they were used to design the system sequentially in order to get the result of the sequential implementation and to note the deferens in the stages of the design between parallel implementation and sequential, the following flowchart shows the proposed system in sequential implementation.

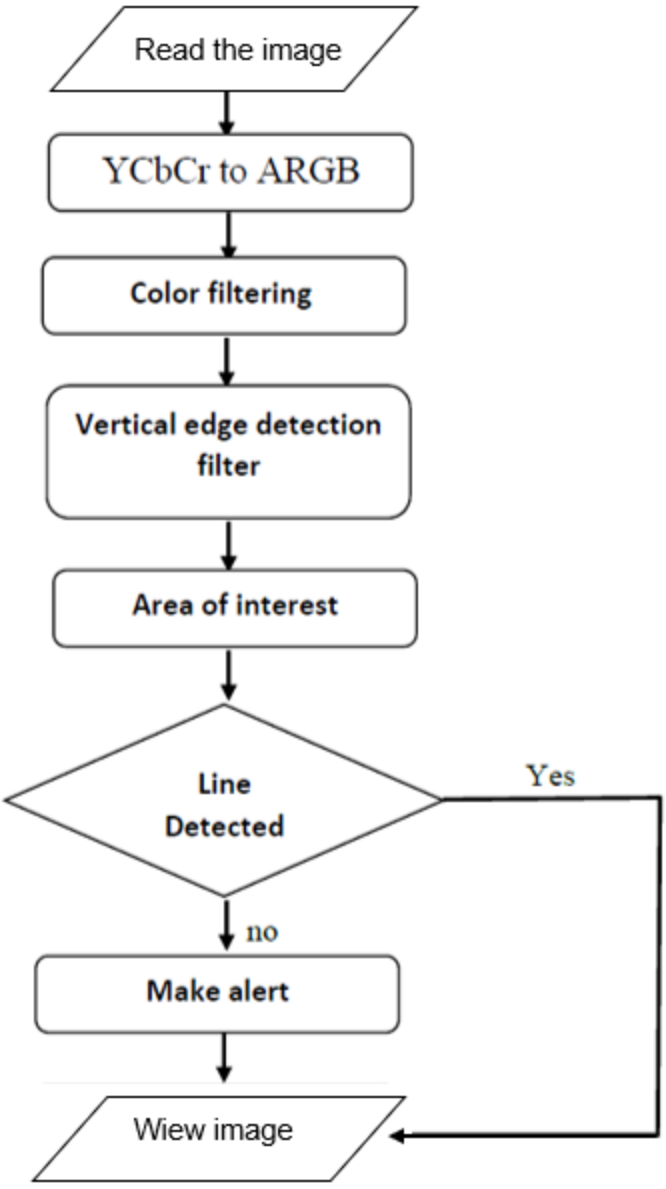

Figure 15. "The proposed system of serial lane detection".

4.1.4 Proposed Parallel Framework

The following stage is determination of possibility and applicability of data parallelism. For a range of tasks on image processing, image matrices carry the data, or it comes on the arrays of lower level 2- dimension. On many occasions, it is possible to divide them into small independent fragments and process them simultaneously, decreasing the time consumed for execution and providing the same result as after sequential processing. In order to detect the image edge, the image matrix can be

divided into small sub-matrices (known as matrix slicing) and initiate sliding a convolution mask to detect edges over each of sub-matrices simultaneously. Nevertheless, when the data to be processed is not that big, not much speed-up will be yielded by applying data parallelism as a consequence of thread overhead. However, as the images made by this application are big, applying data parallelism may grant favorable results. Overall, given k processor cores, the input data of size n should be divided into n/k small fragments and distribute those between k cores, having each core run just one thread (Figure 16).

Figure 16. Matrix slicing.

Several steps are required to perform the parallel computation which are not necessary for the consequential variant of the app (Figure 9). Among the initial major distinctions lies necessity to detect the quantity of threads appropriate for the hardware of phone. Four threads would be suitable to perform the presented function considering that the used phone has four cores. Accordingly, the image then gets divided into four blocks of equal size named sub-image, then share every sub image across one of the 4 available cores. After completing the operation, every core is ready for computing its own chunk of the image. The last step is gathering all the outputs back into single image. The computation outcome is then ready for the following stage: gaining complete control over task management, the Future Task class can also be used, that supports tracking the proceeding of the submitted functions and block up till their full completion. Eventually, the four functions are assigned to an Executor Service that creates thread pool and submits an assigned function to available thread. Using the Executor Service eliminates the problem of thread synchronization and guarantees memory consistency.

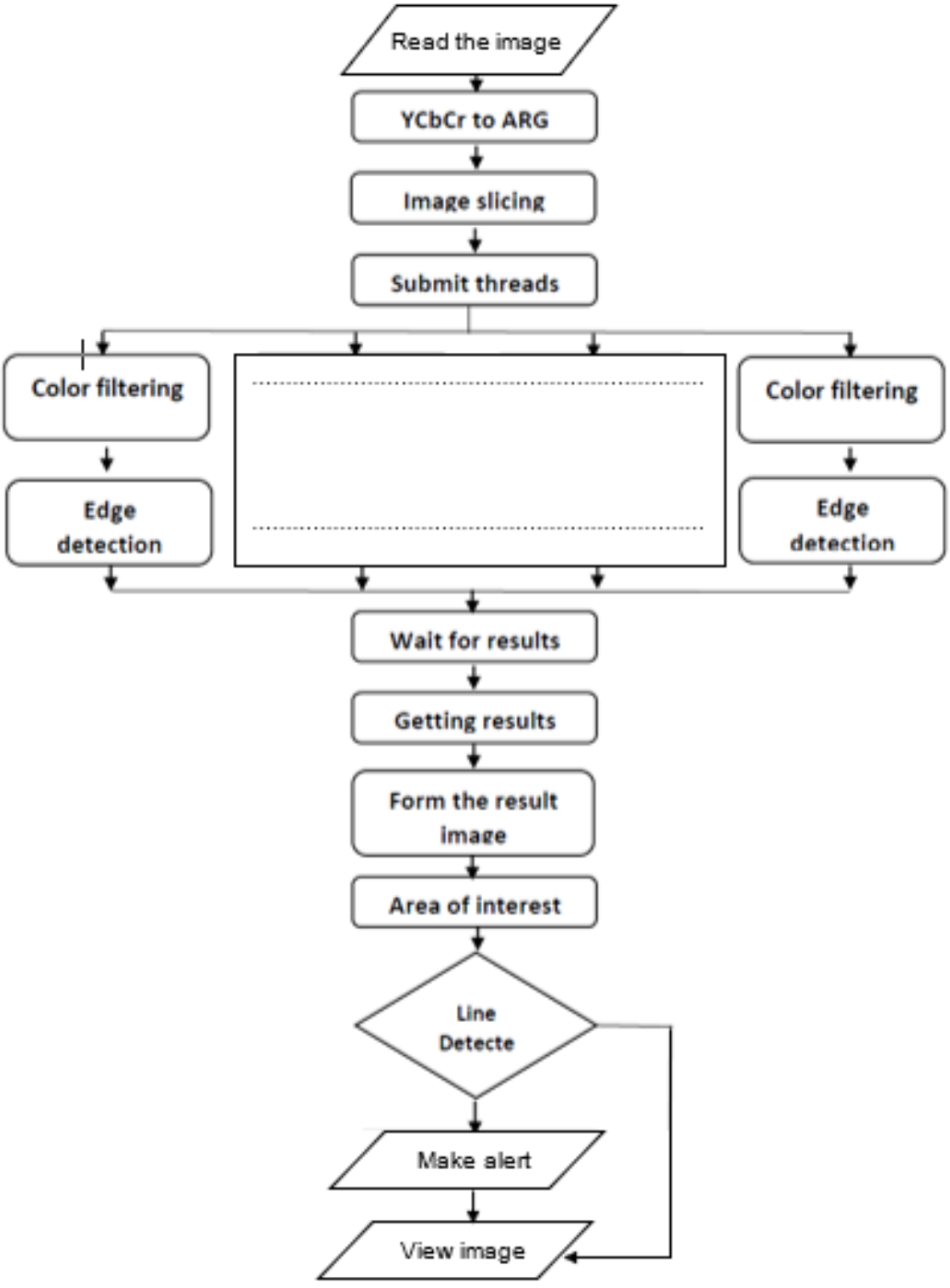

Figure 17. "Modified lane detection system with the proposed parallel framework".

CHAPTER 5

IMPLEMENTION AND RESULTS

5.1 Introduction

In this chapter we will show the steps of the application development and the specification of the devices that we used and how to initiate the development environment then we will test the application and show the results.

5.2 The specification of the devices that we used

We use two smart phones to implement our application, as shown below:

- Samsung Galaxy SIII:

  A: processor type: quad core Coretex A9.

  B. ram memory 1G

  C. screen size 4.8 inches with 720*1280 pixels resolution.

  D. operating system android V4.3(jelly bean)

- Samsung Galaxy S7 edge

  A. processor type: Octa core Exynos 8890

  B. ram memory 4G

  C. screen size 5.1 inches with 1440*2560 pixels resolution.

  D. operating system android V7.0 (Nougat)

5.3 Work environment in Android

The applications of the android system are usually developed in the programming language (java android), and this process is done only using the software development tools for the android system.

5.3.1 Download software development tools for the android system (SDK)

The android system consists of a working platform, a set of tools, code samples as well as documentation for software application development. and it is a software development package that contains collections and functions, it is built as an additional tool of java language development (JDK Java Development Kit), it integrates with the eclipse for the development environment (Integrated Development Environment IDE). the purpose of SDk, is to getting an android simulator through which the android applications are designed and implemented (AVD Android Virtual Device).

5.3.2 Download and install the development environment

The requirements and parts needed to building android applications through a set of steps:

5.3.2.1. The first step: Java SE

At first must be installed the version six or higher of Java SE Java Standard Edition which contains both of JDK and JRE Java Runtime Environment which is downloaded from the site of Sun Microsystem.

5.3.2.2. Second step: Android SDK

Download the applications developers package on the android (Android SDK) from the special site to google developers. after the download completes, the file compression opens and start running an Android simulator manger program (Android Virtual Device Manager AVDM) by this program all the required packages are downloaded via internet connection by developers.

5.3.2.3. Third step: Eclipse

Eclipse: is a multi-use development environment owned by IBM and then a few years ago provided it free and open source and supports platforms through the installation of additions for each platform through the same development environment.

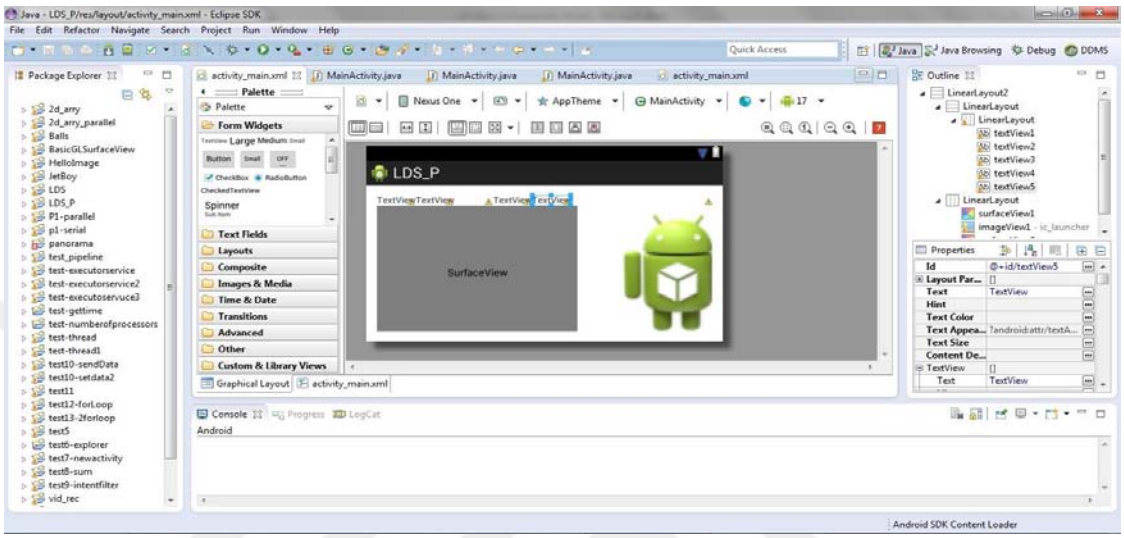

Figure 18. Development Environment (Eclipse).

5.3.2.4. Fourth step: ADT

Download the (ADT Android Development Tools) to support the development of android applications on eclipse. through this addition is connected to the google website to search for files of the application.

Android applications can be built by the development environment (Eclipse) or by (command line). the code for the android applications is usually written in Java and then converted by (DVM Dalvik Virtual Machine), translates as a file (dex Dalvik Executable). and then android package file (APK Android Package), the android SDK includes a tool called DX that converts and translates standard files into direct implementation files.

5.3.3 Running the emulator

Depending on the user's instructions, ADT launches the emulator with default settings and translates the application of the user and executes it.

Figure 19. Emulator for Android system.

5.4 System development

In this section we will show the practical steps of system development.

5.4.1 Camera initialization

In order to use the camera in the smartphone we must define a variable of type Camera, then connect the camera of the phone of this variable so we can control the camera using this variable, then we have to turn on the camera by using the function (camera.open) then set the specifications of the images that will be taken from the camera such as the high and the width of the image and the image orientation (vertical or horizontal) or were the image will be shown .in order to do all of the above we built a function called initcamera, we can summarize the job of this function by the following:

A. Turn on the camera
B. Set the orientation of the image to landscape
C. Define a parameter of type camera, parameters by using this parameter we will reach to the camera specifications
D. Set the height and width of the image.
E. Define where the image will be shown.

5.4.2 Data acquisition

In order to get the data of each frame coming from the camera we built a function called on preview frame which returns the data of each frame.

As we mentioned earlier, this data come as one dimensional array and its elements of type byte, in order to use this data we built a function called gety which transforms y from byte to integer then rearrange these values in a shape of number of two dimensional matrix's which represent the image that will be processed, this images will be gray because we used only the value of y in creating it.

Figure 20. Set of images before and after performing the processing.

The images on the left are before processing and the images on the right are after processing.

5.4.3 Define the area of the image that will be processed

There are areas in the image where no lines will be, they are upper part of the image which will be the sky and the lower part of the Image which will be the front of the car to eliminate these part we built a function to cut those parts so no processing will be done there by doing that we reduced the amount of data that will be send to the next steps which leads to increase the processing speed.

Figure 21. A set of images before determining the processing areas.

5.4.4 Recognize the road lines using the colors

In this stage we applied threshold on the image to extract the high value of the colors which can be road line, to do that we built a function that lets the value more than val pass and remove the value which is less than val. We calculated the value of val after doing a lot of tests on a big number of images and we conclude the value that gave as the best results was 180 so that the value of val set to 180

Figure 22. Set of images before and after applied threshold.

The images on the left are before applied threshold and the images on the right are after applied threshold.

5.4.5 Recognize the road lines using shape

In this stage we applied a suggested filter in the image that extract the road lines, to do that we built a functions that applies the filter on the image (convolution).at the beginning we must built the suggested filter and apply it according to the equation mentioned in chapter 4 , then apply threshold on the image. The value of val2 was set after doing many test and the value gave as best results was 100.

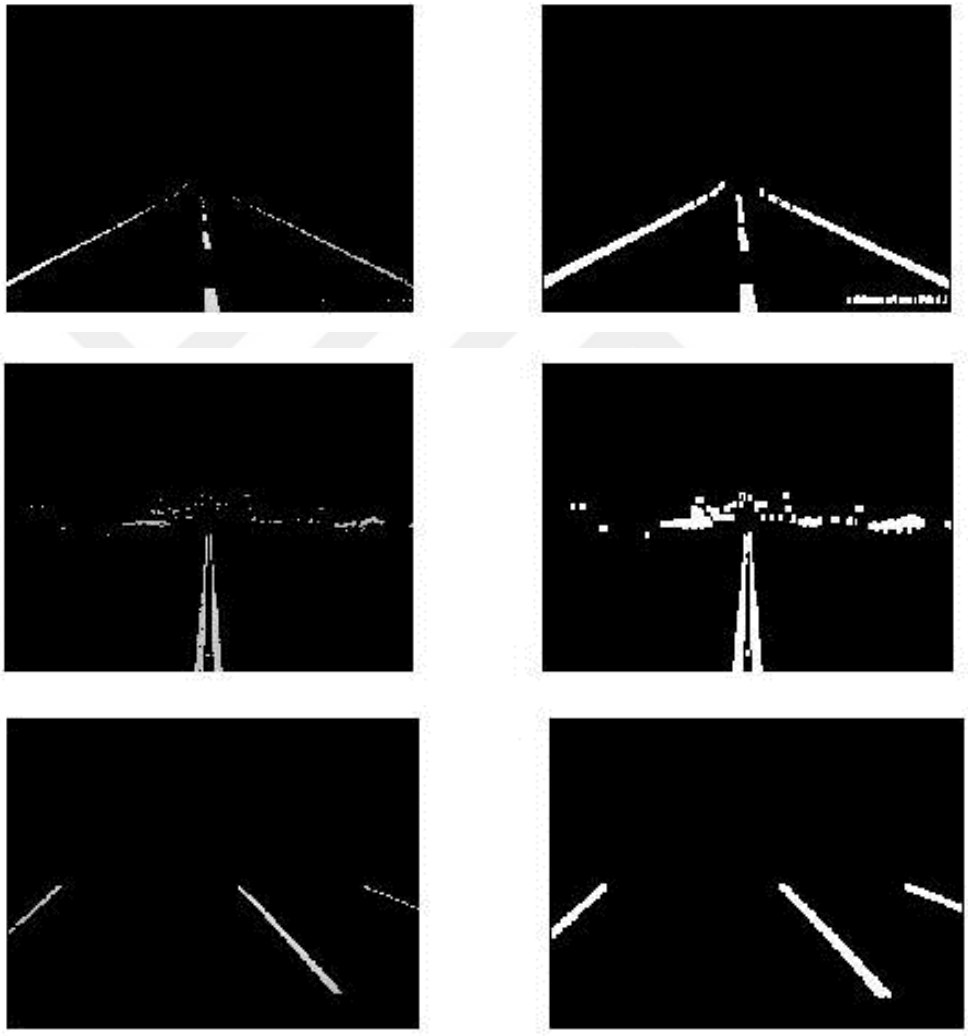

Figure 23. Set of images before and after applied a suggested filter.

The images on the left are before applied the suggested filter and the images on the right are after applied the suggested filter.

- Comparing the results of the suggested filter with the results of some of the standard filters.

We applied the suggested filter and some of the standard filters on a number of the images, the suggested filter gave as the best results because it showed the hole line not just the borders like in the standards filters.

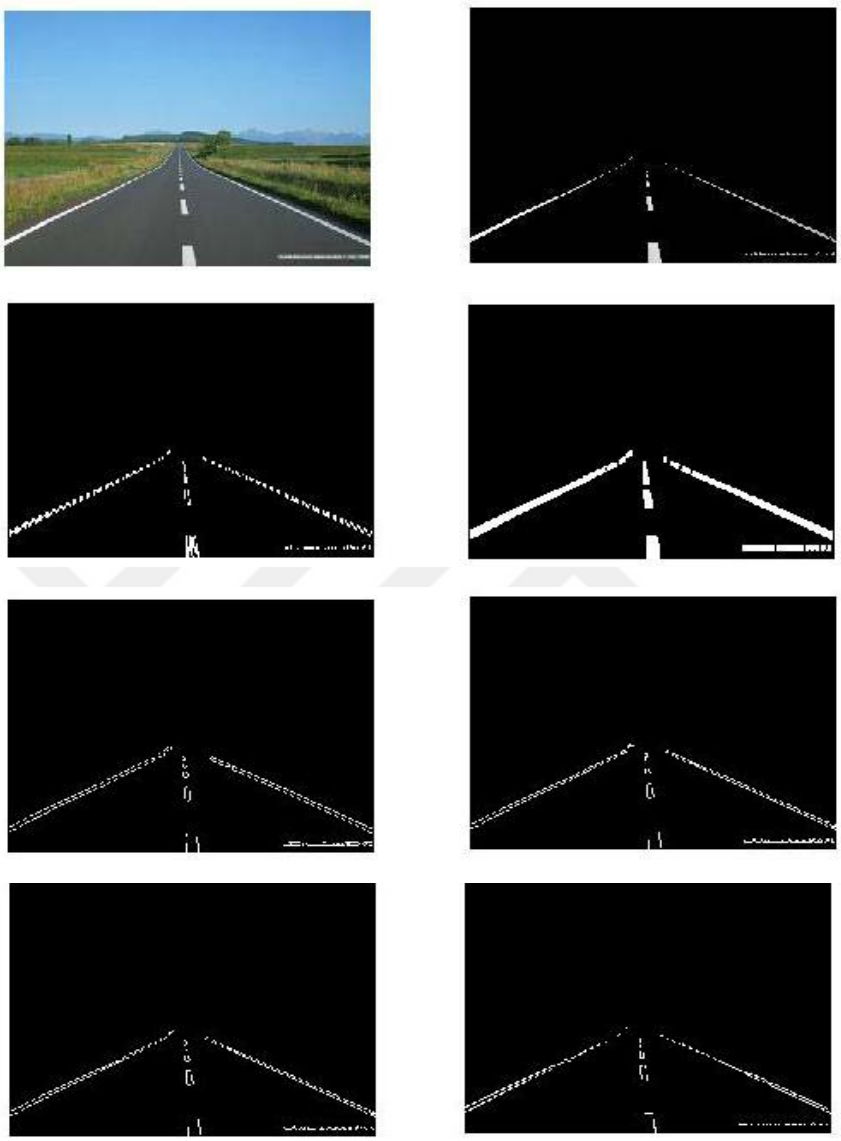

Figure 24. Comparison the results of the proposed filter with some of the standard filters.

5.4.6 Determine the position of the car in relations to the road line

To help the driver to place the phone in the correct place in the car, we highlighted an area that if the car is in the correct place in the lane that area must be empty and the driver must place the phone in a way that this area still empty.

During the work of the system, the system scans that area to find out if there is any line, if there is a line in that area that implies the car to be leaving the lane for the system to fire the alarm.

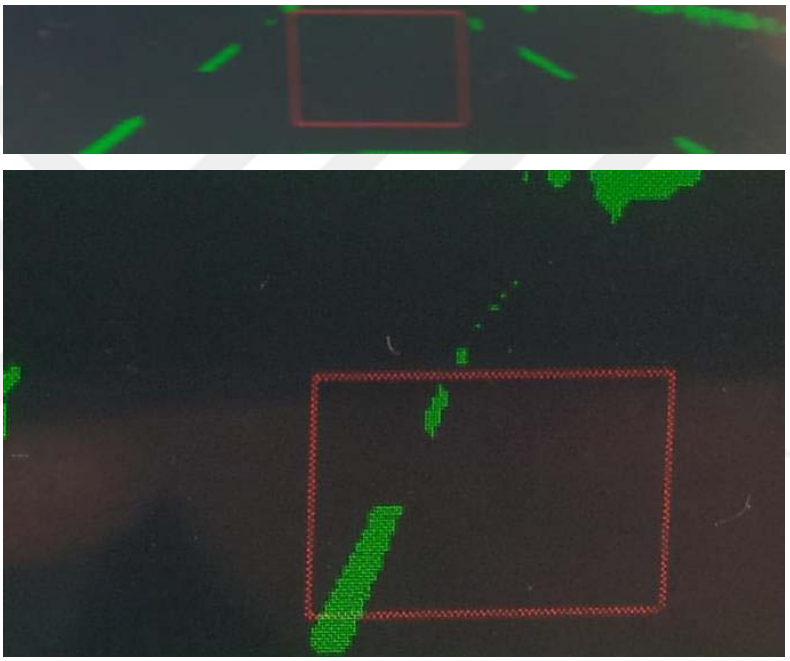

Figure 25. The process of selecting the area to scan for lines.

5.4.7 Apply the principle

We implemented our application in many different scenarios, we used two resolutions (320*240 and 640*480), and we implemented our application sequentially and using two, four, and eight threads in parallel.

The results (execution time in seconds) of the implementation on the Samsung galaxy SIII are shown in the table below:

Table 2. Average execution time using mobile (SIII) with quad core.

Resolutions	Sequential	2 threads	4 threads	8 threads
320*240	11.043	7.003	6.506	6.713
640*480	43.477	30.970	28.045	29.994

After getting the execution times, we calculated speedup and efficiency and as shown below:

Table 3. The value of speedup and efficiency using (SIII) with quad core.

Resolutions		2 threads	4 threads	8 threads
320*240	Speedup	1.57	1.69	1.64
	Efficiency	0.39	0.42	0.41
640*480	Speedup	1.4	1.55	1.44
	Efficiency	0.35	0.38	0.36

After studying and analyzing the results in the tables above, we concluded the following:

- We get the best processing times, speedup, and efficiency when the number of the threads was equal to the number of the cores of the phone processor, that means the best speedup is 1.69, and the best efficiency is 0.42.

- The processing time using 8 threads was better than the processing time using 2 threads in the quad core smartphones, because when we used 2 threads in the quad core smartphone that lidded to using 2 cores and leaving 2 cores with no work (bad utilization), but when we used 8 threads that lidded to utilize all of the 4 cores (this case called multithreading on each core).

- When the number of the threads was higher than the number of the cores, the processing time increased, the efficiency and speedup where decrease, that is because the processor needed more time to move from one threads to another, this called Context Switching.

The results (execution time in seconds) of the implementation on the Samsung galaxy S7 edge are shown in the table below:

Table 4. Average execution time using mobile (S7) with octa core.

Resolutions	Sequential	2 threads	4 threads	8 threads
320*240	5.0334	4.886	4.231	4.143
640*480	6.634	5.804	5.342	4.342

After getting the execution times, we calculated speedup and efficiency and as shown below:

Table 5. The value of speed up and efficiency using mobile (S7) with octa core.

Resolutions		2 threads	4 threads	8 threads
320*240	Speedup	1.03	1.18	1.21
	Efficiency	0.12	0.14	0.15
640*480	Speedup	1.14	1.24	1.52
	Efficiency	0.14	0.15	0.19

After studying and analyzing the results in the tables above, we concluded the data from the main memory and this needs time. following:

- We get the best processing times, speedup, and efficiency when the number of the threads was equal to the number of the cores of the phone processor, that means the highest speedup is 1.52, and the highest efficiency is 0.19.

- We get the best processing time and speed up, by using 8 threads by Samsung galaxy S7 edge, the reason is due to the large size of memory(RAM) in the device, which led to increase in the size of data that can be transferred from external memory to the main memory to be available to the processor when needed, but if the size of memory is not enough all the data then must be bring the new

The figure below shows the system working in case of save driving and in case of warning:

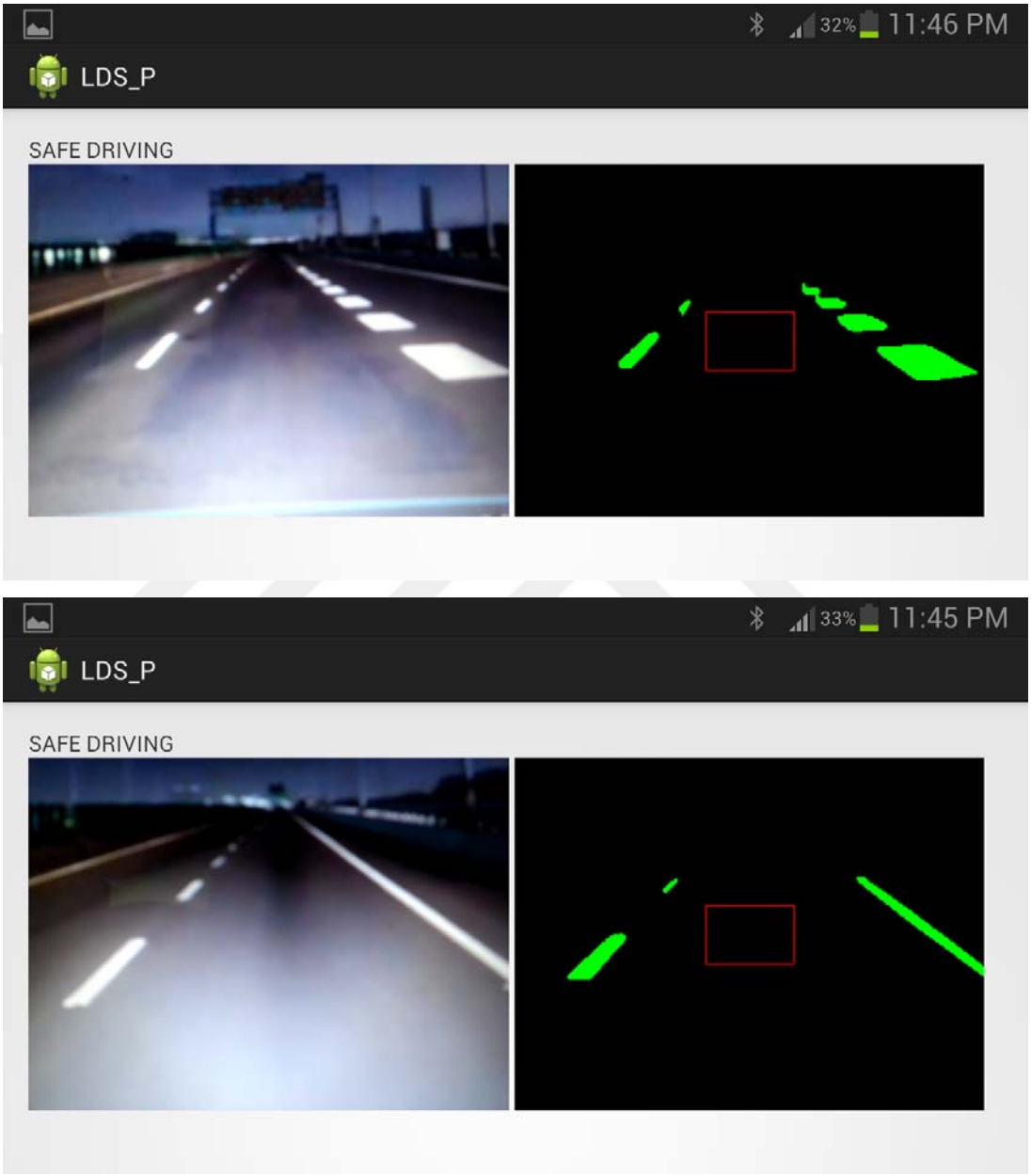

Figure 26. The system working in case of save driving.

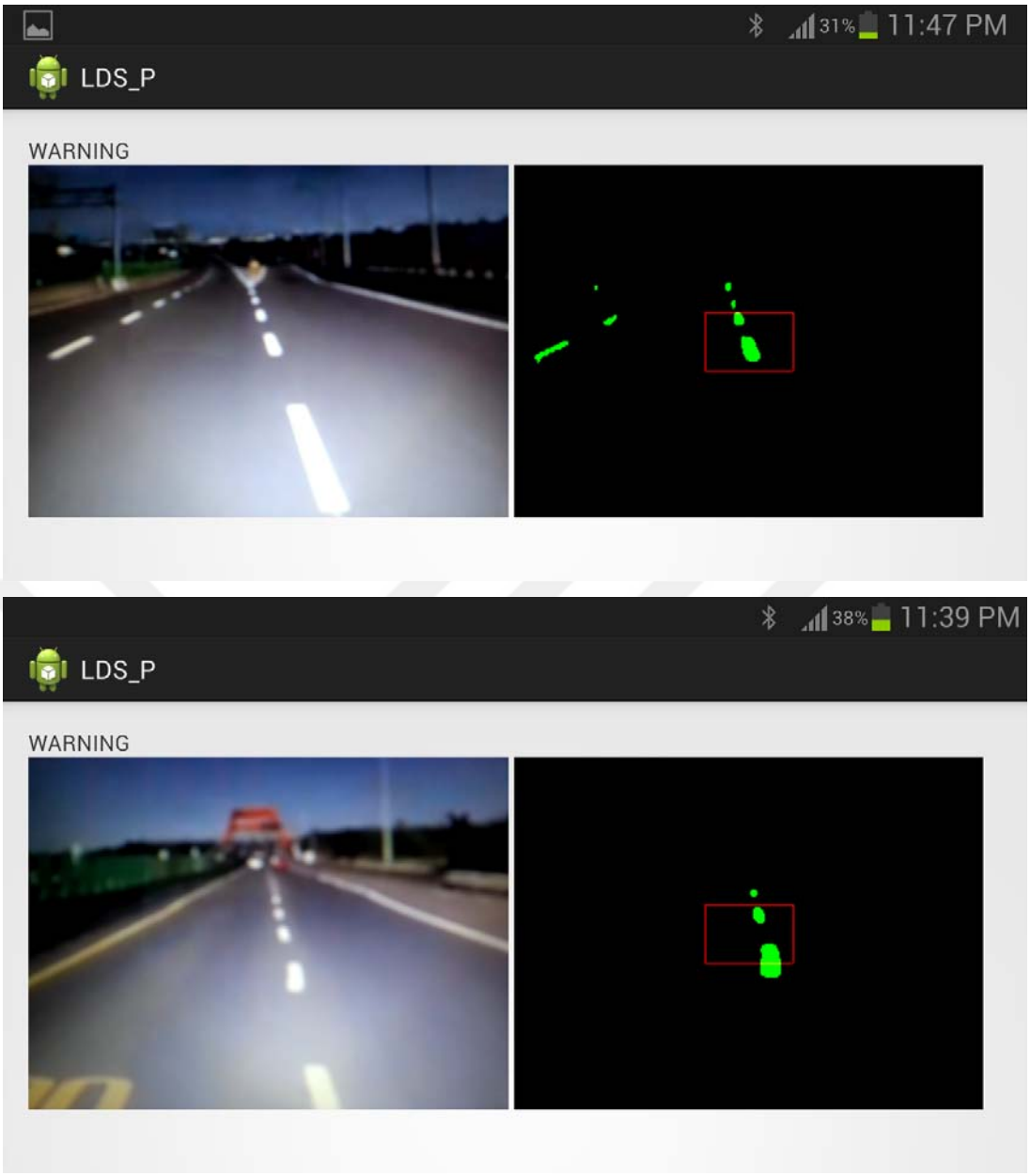

Figure 27. The system working in case of warning.

CONCLUSION

After studding and analyzing the results in the table above, we concluded the following:

1. We get the best processing times, speedup, and efficiency when the number of the threads was equal to the number of the cores of the phone processor.
2. The processing time using 8 threads was better than the processing time using 2 threads in the quad core smartphones, because when we used 2 threads in the quad core smartphone that lidded to using 2 cores and leaving 2 cores with no work (bad utilization), but when we used 8 threads that lidded to utilize all of the 4 cores (this case called multithreading on each core) (SIII).
3. When the number of the threads was higher than the number of the cores, the processing time increased, the efficiency and speedup where decrease, that is because the processor needed more time to move from one threads to another, this called Context Switching (SIII).
4. The processing time when the number of threads is 2 better than the processing time when using 4 threads, and the last one is better than the processing time when using 8 threads, in dual-core processors phones, the excess time is the time spent in the process of switching from one thread to another (Context Switching).
5. We get the best processing time and speed up, by using 8 threads by Samsung galaxy S7 edge, the reason is due to the large size of memory(RAM) in the device, which led to increase in the size of data that can be transferred from external memory to the main memory to be available to the processor when needed, but if the size of memory is not enough all the data then must be bring the new data from the main memory and this needs time.

REFERENCES

[1] El-Rewini H. and Abd-El-Barr M. et al., 2005, "Introduction to Advanced Computer Architecture and Parallel Processing", John Wiley & Sons, Inc. ISBN: 978-0-471-46740-3.

[2] Nancy J. V., Richard J. F., James A. M. et al., 2011, " PARALLEL PROCESSING: THE NEXT GENERATION OF COMPUTERS", National Energy Technology Laboratory.

[3] Evangelinos C. and Hill C. et al., 2008, "Cloud Computing for parallel Scientific HPC Applications: Feasibility of running Coupled Atmosphere-Ocean Climate Models on Amazons EC2." ratio, vol. 2, no. 2.40, pp. 2–34, 2008.

[4] Sarita V. A., Vikram S. A., et al., 2008, " Parallel Computing Research at Illinois The UPCRC Agenda", The Board of Trustees of the University of Illinois.

[5] Mohr B. et al., 2006, " Introduction to Parallel Computing", Copyright John von Neumann Institute for Computing.

[6] Lewis, T. G. and El-Rewini, H. et al., 1992, "Introduction to Parallel Computing", Prentice-Hall.

[7] Hwang K. et al., 1993, "Advanced Computer Architecture: Parallelism, Scalability and Programmability", McGraw-Hill, Inc. ASIN: 7111067126.

[8] Parhami B. et al., 2002, "Introduction of Parallel Processing: Algorithms and Architectures", Kluwer Academic Publishers, New York, Boston, Dordrecht, London, Moscow. All rights reserved.

[9] Fritz N. et al., 2009, "SIMD Code Generation in Data-Parallel Programming", epubli.uni-saarland.

[10] Sung M. et al., 2000, " SIMD Parallel Processing", 6.911: Architectures Anonymous.

[11] Quammen C. et al., 2002, "Introduction to Programming Shared-Memory and Distributed-Memory Parallel Computers", ACM Crossroads.

[12] Zargham M. et al., 1996, "Computer Architecture: Single and Parallel Systems", Prentice-Hall. ISBN-10: 0130106615, ISBN-13: 978-0130106612.

[13] Hennessy J. and Patterson D.et al., 1996, "Computer Architecture: A Quantitative Approach", Morgan Kaufmann Publishers, SF, CA.,1996, ISBN-10: 1558605967 | ISBN-13: 978-1558605961.

[14] Marshall D., 2011, "Parallel Programming with Microsoft Visual Studio", Microsoft Corporation by: O'Reilly Media.

[15] Feitelson D. and Rudolph L., 1996, "Job scheduling strategies for parallel processing", Springer- Verlag Berlin Heidelberg, ISBN 3-540- 61864-3.

[16] Tokhi M. et al., 2003, "Paralle Computing for Real-time Signal Processing and Control", Springer- Verlag London Limited, ISBN 1- 85233-599-9.

[17] Conder S. and Darcey L., 2011, "Android Wireless Application Development", Second Edition, Addison-Wesley, ISBN-13: 978-0-321- 74301-5.

[18] Guihot H., 2012, "Pro Android Apps Performance Optimization", Apress, ISBN-13: 978-1-4302-4000-6.

[19] Laurence T. Yang, Daniel C. Doolan et al., 2006, "Mobile Parallel computing", Proceedings of The Fifth International Symposium on Parallel and Distributed Computing, IEEE International.

[20] Rudra N. Hota et al., 2009, "A Simple and Efficient Lane Detection using Clustering and Weighted Regression", 15th International Conference on Management of Data COMAD, Mysore, India.

[21] A. LÓPEZ et al., 2010, "Robust Lane Markings Detection and Road Geometry Computation", International Journal of Automotive Technology, Vol. 11, No. 3, pp. 395−407. Conder S. and Darcey L., 2011, "Android Wireless Application Development", Second Edition, Addison-Wesley, ISBN-13: 978-0-321- 74301-5.

[22] Bryan Catanzaro et al., 2010, “Ubiquitous Parallel Computing form Berkeley, Illinois, and Stanford”, IEEE Computer Society, pp. 41-55.

[23] Panagiotis Tsogkas, 2010, “Evaluating Skandium’s Divide-and-Conquer Skeleton”, Master Thesis, School of Information, University of Edinburgh

[24] Emmanouil Koukoumidis et al., 2012, “Leveraging Smartphone Cameras for Collaborative Road Advisories”, IEEE TRANSACTIONS ON MOBILE COMPUTING, VOL. 11, NO. 5.

[25] Panya Chanawangsa and Chang Wen Chen, 2012" A New Smartphone Lane Detection System: Realizing True Potential of Multi-Core Mobile Devices" IEEE MoVid'12, Chapel Hill, North Carolina, USA.

[26] USA Ding-Yun Chen et al., 2013, " MVSE: A Multi-Core Video Decoder System Level Analytics Engine ", IEEE International Symposium on VLSI Design, Automation, and Test (VLSI-DAT).

[27] Kangwoon Lee et al., 2013, "CPU and GPU Parallel Processing for Mobile Augmented Reality ", IEEE 6th International Congress on Image and Signal Processing (CISP).

[28] Characterization Zongwei Zhu et al., 2014, "A Thread Behavior-based Memory Management Framework on Multi-core Smartphone", IEEE 19th International Conference on Engineering of Complex Computer Systems.

[29] Kuei-Chung Chang et al., 2014, " Memory Behavior Profiler for Android Applications" IEEE 3rd Global Conference on Consumer Electronics (GCCE)

[30] Po-Hsien Tseng et al., 2014, " User-Centric Energy-Efficient Scheduling on Multi-Core Mobile Devices" IEEE DAC’14, San Francisco, CA

[31] Qianao Ju et al., 2015, " Benchmarking Render Script: Potential for Energy Efficient Multi-Core Mobile Devices", IEEE.

[32] Wonik Seo et al., 2015, " Big or Little: A Study of Mobile Interactive Applications on an Asymmetric Multi-Core Platform", IEEE International Symposium on Workload.

[33] Alejandro Acosta and Francisco Almeida, et al., 2015 " Parallel implementations of the Particle Filter algorithm for Android mobile devices" IEEE Parallel implementations of the Particle Filter algorithm for Android mobile devices.

[34] Andreas Kurth et al.,2016 " Mobile Ultrasound Imaging on Heterogeneous Multi-Core Platforms" IEEE ESTIMedia'16, Pittsburgh, PA, USA.

[35] Berstein A. J.,1966 "Analysis of programs for parallel processing" IEEE Transactions on electronic computers, vol, ec.15, No.5