The International Journal Of Engineering And Science (IJES)
|| Volume || 6 || Issue || 3 || Pages || PP 86-92 || 2017 ||
ISSN (e): 2319 – 1813 ISSN (p): 2319 – 1805

A General-Purpose Architectural Approach to Energy Efficiency for Greendroid Mobile Application Processor

Andemariam Mebrahtu[1], Balu Srinivasulu[2]
[1]*Lecturer, Computer Science, Eritrea Institute of Technology, Eritrea, North East Africa.*
[2]*Lecturer, Computer Science, Eritrea Institute of Technology, Eritrea, Nortth East Africa.*

--**ABSTRACT**--
Mobile application processors are soon to replace desktop processors as the focus of innovation in microprocessor technology. Already, these processors have largely caught up to their more power hungry cousins, supporting out-of order execution and multicore processing. In the near future, the exponentially worsening problem of dark silicon is going to be the primary force that dictates the evolution of these designs. We have argued that the natural evolution of mobile application processors is to use this dark silicon to create hundreds of automatically generated energy-saving cores are called conservation cores, which can reduce energy consumption by an order of magnitude. Conservation cores(C-cores) try to solve utilization wall and consequently Dark Silicon issues.Greendroid is a development library for the android platform. It is intended to make UI developments easier and consistent through your applications. This paper describes Greendroid, a research prototype that demonstrates the use of such cores to save energy broadly across the hotspots in the android mobile phone software stack.

Keywords: *Greendroid, prototype, UI, C-cores, Dark Silicon.*

Date of Submission: 02 March 2017 Date of Accepted: 20 March 2017

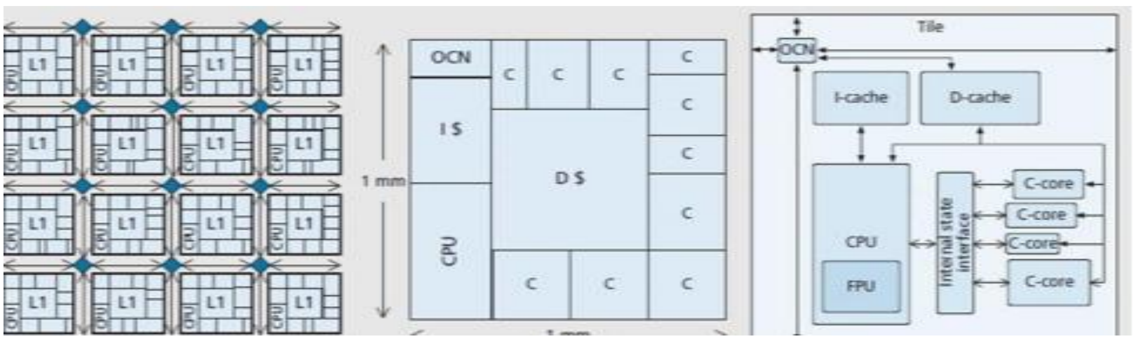
I. INTRODUCTION

With the advancement in electric design industry normal mobile phones with only calling capabilities have gone obsolete now. Mobile phones are now replaced by smart phones which run on a open source operating system like android or iOS. Smart phones have integrated capabilities of personal digital assistant, music player, digital camera and a GPS based navigation device. A Greendroid processor will have many smart conservation core also pronounced as c- cores. Each core targets a specific portion of the android operating system. Android operating system is well suited for use with c-cores. These cores are re-configurable.
Greendroid is a mobile processor which will reduce power consumption in smart phones. Greendroid will provide many specialized processors targeting key portions of android based smart phone.
Greendroid will reduce power consumption for these codes by making use of special computing cores known as conservation core. This mobile application processor is based on 45 nm multi core technology and can accomplish general purpose mobile applications with 11 times less energy than the best available power efficient design in the market, at similar or better performance levels.it does this through the use of a number of automatically generated highly specialized, sophisticated power optimized cores also called as conservation cores [1].
A direct consequence of this is dark silicon—large swaths of a chip's silicon area that must remain mostly passive to stay within the chip's power budget.
Dark silicon is when you have three billion transistors on your chip but you can only use 1.8 percent of them at a time, so as to squeak under the threshold of your chip's draconian energy budget.
Currently, only about 1 percent of a modest-sized 32-nm mobile chip can switch at full frequency within a 3-W power budget.
This paper describes a key technological problems and solutions for microprocessor architects, which we call the utilization wall. The utilization wall says that, with each successive process generation, power constraints cause the percentage of a chip that can actively switch to drop exponentially [2].

II. ARCHITECTURE OF GREENDROID

A Greendroid processor combines general-purpose processors with application-specific coprocessors that are very energy efficient. These conservation cores, or c-cores, execute most of an application's code and will account for well over 90 percent of execution time. Green-Droid is a heterogeneous tiled architecture. It contains an energy-efficient 32-bit 7-stage in order pipeline that runs at 1.5 GHz in a 45 nm process technology.

It includes a single-precision floating point unit (FPU), multiplier, 16- k byte I-cache, translation look aside buffer (TLB), and 32-kbyte banked L1 data cache [3].
The architecture also includes a mesh-based on-chip network (OCN). The OCN carries memory traffic and supports fast synchronization primitives illustrates how it uses a grid-based organization to connect multiple tiles.
The execution starts on one of the CPU'.When the CPU recognizes the hot code transfers the execution on the appropriate c-core. Execution moves from tile to tile with respect the availability and their specialization availability and their specialization. Data associated with a given c-core usually resides in the associated L1 cache resides in the associated L1 cache. C-cores largely transparent to developers.



(a) (b)

Figure 1 . The Greendroid architecture. The system comprises 16 non identical tiles (a) Array of Tiles (b) One Tile(c) Interface between CPU and C Core.

III. ORIGIN OF DARK SILICON

Conventional chip designs in a mobile phone lose significant energy due to the presence of unused transistors called dark silicon.
Dark silicon is when we have three billion transistors on our chip but we can only use 1.8 percent of them at a time, so as to squeak under the threshold of our chip's draconian energy budget. So the lights are out on vast swaths of a chip's area; hence "dark silicon".
As the transistor counts are growing faster, the underlying energy efficiency is improving; a direct consequence of this is the phenomenon of dark silicon.
The dark silicon problem is directly responsible for the desktop processor industry's decision to stop scaling clock frequencies and instead build multi-core processors [5].
It plays an equally pivotal role in shaping the future of mobile processors as well. With each process generation, dark silicon is a resource that gets exponentially cheaper, while the power budget becomes exponentially more valuable in comparison.
Dark silicon is necessary, because engineers are unable to reduce chips' operating voltages any further to offset increases in power consumption and waste heat produced by smaller, faster chips.
This dark silicon limits the utilization of the application processors to the fullest and it can be called the utilization wall: With each successive process generation, power constraints cause the percentage of a chip that can actively switch to drop exponentially [4].
The utilization wall is changing the way people build chips. To make this point, whose "turbo mode" makes some cores run faster while the remaining cores are switched off?

3.1. The solution: C-cores

Greendroid proves to be a boon to this dark silicon problem. It follows the ideology "If you fill the chip with highly specialized cores, then the fraction of the chip that is lit up at one time can be the most energy efficient for that particular task". These highly specialized cores are known as conservation cores or c-cores.
They sit alongside the general-purpose processor and share the same data cache and memory hierarchy and is divide up code into two types: **Cold code and Hot code**

3.1.1: Cold code which is code that's infrequently used, runs on the general purpose processor as before.

3.1.2: Hot code by contrast, which is accessed frequently, is diverted to the C-cores to reduce energy use.
"Conservation Cores: Reducing the Energy of Mature Computations,

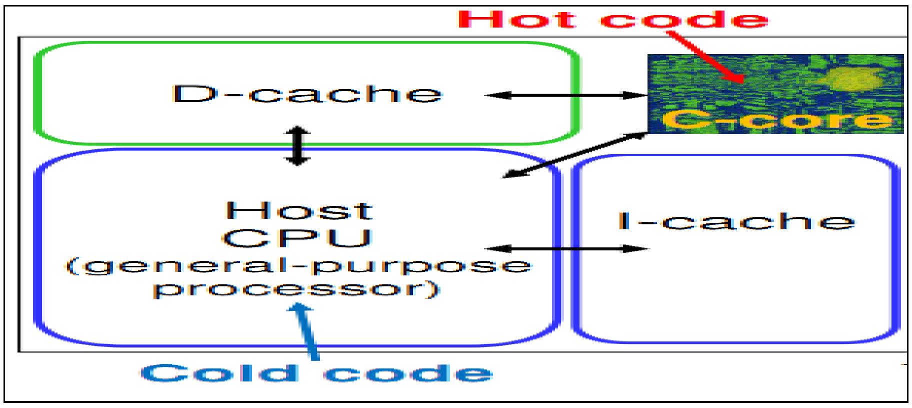


Fig2. Cold code and Hot Code

Specialized, energy-efficient processors can increase parallelism by reducing the per-computation power requirements and allowing more computations to execute under the same power budget.
To pursue this goal Conservation cores or c-cores, are specialized processors that focus on reducing energy and energy-delay instead of increasing performance.
This focus on energy makes c-cores an excellent match for many applications that would be poor candidates for hardware acceleration (e.g., irregular integer codes).
A tool chain is used for automatically synthesizing c-cores from application source code and demonstrating that they can significantly reduce energy and energy-delay for a wide range of applications. The c-cores support patching, a form of targeted reconfigurability, that allows them to adapt to new versions of the software they target [6].The C- core Life Cycle and Main ideas behind c-cores**.**

IV. THE C-CORE LIFE CYCLE AND MAIN FEATURES

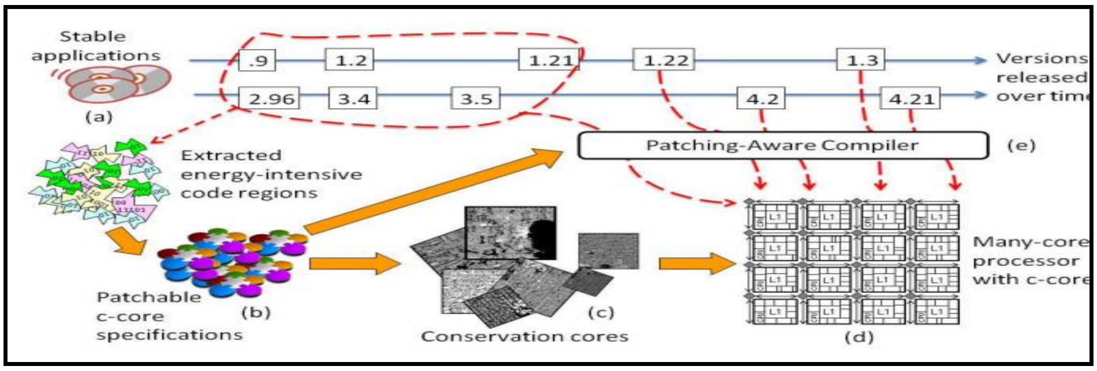

Stable applications
(a)
.9
1.2
1.21
1.22
1.3
2.96
3.4
3.5
4.2
4.21
Versions released over time
Extracted energy-intensive code regions
Patching-Aware Compiler
(e)
Patchable c-core specifications
(b)
(c)
Conservation cores
Many-core processor with c-cores
(d)

Fig3. C-core Life Cycle

4.1: C-cores details
- ❖ We must do a profiling of the code
- ❖ We must do a profiling of the code
- ❖ A specialized circuit (c-core) tries to mirror the hot code adding an extra logic that the hot code adding an extra logic that allows patching
- ❖ Cold code runs on the CPU
- ❖ Specialized compiler is responsible to recognize what code aligns with the c-cores
- ❖ We also have a runtime system that manages the allocation of c-cores according to availabilty

4.2: Data path
- ❖ Functional units (adders, shifters) to execute instructions
- ❖ Multiplexer to implement control decisions
- ❖ Registers

4.3: Control unit
- ❖ Implements the state machine that mirrors the ◦ Implements the state machine that mirrors the
- ❖ Control Flow Graph
- ❖ Track s branch outcomes (computed in data path) to determine witch hardware block must be active

4.4: Synthesizing c-cores

- ❖ The design of the c cores is not done by hand.
- ❖ A C/C++-to-Verilog tool chain is used to convert the code in hardware
- ❖ The tool chain identifies the main loops and functions given a target workload
- ❖ The CFG and the data control flow graph are created.
- ❖ The compiler generates
- The **verilog** code for the control unit
- The data path that closely mimics the representations.
- Function stubs that applications can call in place of the original functions to invoke the hardware.
- Description of the c-core, used when we update the function.
- ❖ Small changes in source code correspond to small changes in hardware changes in hardware.
- ❖ Since the target is to minimize the energy consumption and not to achieve better performance we can exploit many more C constructs than when we try to get more parallelism in the code [7].

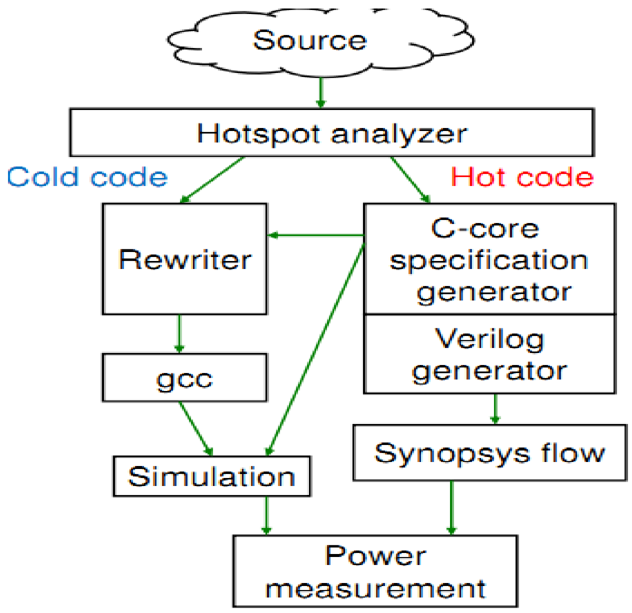


Fig.4 Core compiler generator

4.5: Patching

- ❖ Since the software evolves the c-core must y Since the software evolves the c-core must adapt too

Redifine compile time constants in hardware

Exception mechanism that allows to transfer the control back and forth the CPU and the c-cores control back and forth the CPU and the c cores

- ❖ The area of the chip is increased
- ❖ But the experiments show that the y But the experiments show that the adaptation process can hold for a decade
- ❖ Remember that the mean life cycle of a smart phone is 3 years

4.6: Accelerators

- ❖ The main part of specialized hardware is used to The main part of specialized hardware is used to achieve better performance
- ❖ We need simple code that exposes parallelism and a simple way to create a circuit that is neither costly nor complex
- ❖ In GreenDroid accelerators are mainly used to reduce energy consumption
- ❖ More code can be suitable to create a c-core that executes it
- ❖ First we accelerate the code that can be parallelized, then we take the remaining code and we try to map it to c-cores as much as possible.

4.7: High level synthesis tools

- ❖ Since the code is different and less parallelizable we must have a completely automatic tool chain
- ❖ We can't have user aided process because

- Code is too large
- Code is constantly evolving
- HLS supports I/O and system calls
- Also parts of the kernel are translated

4.8: Constructing a C-core

❖ C-cores start with source code
- Can be irregular, integer programs
- Parallelism-agnostic

❖ Supports almost all of C:
- Complex control flow

e.g., goto, switch, function calls
- Arbitrary memory structures

e.g., pointers, structs, stack, heap
- Arbitrary operators

e.g., floating point, divide
- Memory coherent with host CPU

```
SumArray(int *a, int n)
{
int i = 0;
int sum = 0;
for (i = 0; i < n; i++) {
sum += a[i];
}
return sum;
}
```

V. ANDROID: GREENDROID’STARGET WORKLOAD-ANDROID™

Android is an excellent target for a c-core enabled GreenDroid-style architecture. Android comprises three main components: a version of the Linux kernel, a collection of native libraries (written in C and C++), and the Dalvik virtual machine (VM). Android’s libraries include functions that target frequently executed, computationally intensive (or “hot”) portions of many applications (e.g., 2D compositing, media decoding, and garbage collection).

- Google’s OS + app. environment for mobile devices.
- java applications run on the Dalvik virtual machine
- Apps share a set of libraries (libc, OpenGL, SQLite, etc.)

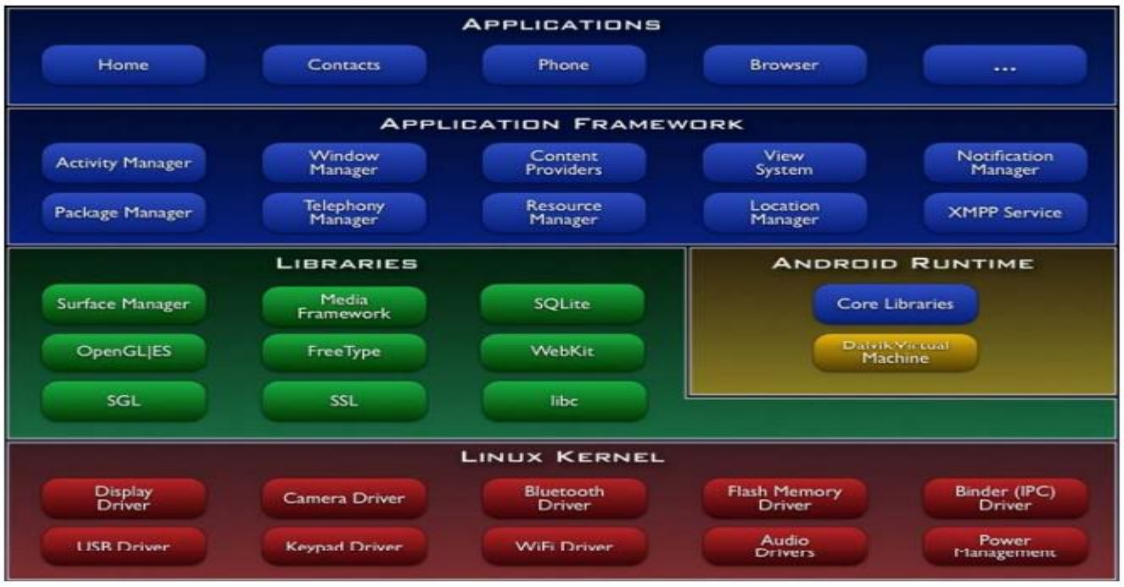


Fig.5 Android Application stack

The remaining “cold” code runs on the Dalvik VM, and, as a result, the Dalvik VM is also “hot.” Therefore, a small number of c-cores that target the Android libraries and Dalvik VM should be able to achieve very high coverage for Android applications.

We have profiled a diverse set of Android applications including the web browser, Mail, Maps, Video Player, Pandora, and many other applications. We found that this workload spends 95 percent of its time executing just 43,000 static instructions. Our experience building c-cores suggests that just 7 mm of conservation cores in a 45 nm process could replace these key instructions [8].

Even more encouraging, approximately 72 percent of this 95 percent was library or Dalvik code that multiple applications used.

Android's usage model also reduces the need for the patching support c-cores provide. Since cell phones have a very short replacement cycle (typically two to three years), it is less important that a c-core be able to adapt to new software versions as they emerge.

5.1 Applying C-cores to Android

❖ Android is well-suited for c-cores
- Core set of commonly used applications
- Libraries are hot code
- Dalvik virtual machine is hot code
- Libraries, Dalvik, and kernel & application hotspots - c-cores
- Relatively short hardware replacement cycle

5.2 GreenDroid: Applying Massive Specialization to Mobile Application Processors

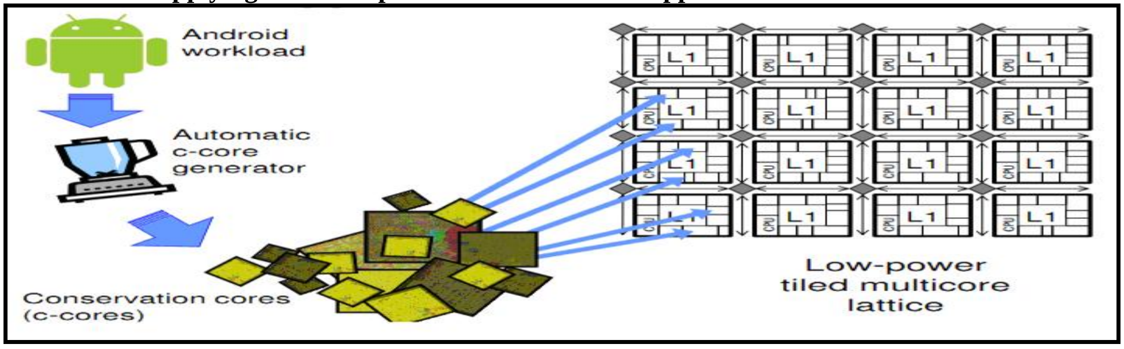

Android workload
Automatic c-core generator
Conservation cores (c-cores)
Low-power tiled multicore lattice
L1
CPU

Fig 6.C-core low power tiled multi core architecture

❖ Software may change – HW must remain usable
- C-cores unaffected by changes to cold regions

❖ Can support any changes, through patching
- Arbitrary insertion of code – software exception mechanism
- Changes to program constants – configurable registers
- Changes to operators – configurable functional units

❖ Software exception mechanism
- Scan in values from c-core
- Execute in processor
- Scan out values back to c-core to resume execution

VI. BENEFITS OF GREENDROID

6.1: Try to make Android applications alike: The openness of the Android operating system makes it less coherent. Seeing so many applications that use totally different ergonomics or designs drives me mad. Indeed, I truly think, Android applications need coherency in order to get more and more users around the world. GreenDroid is kind of a try to bring coherency to your applications and therefore to the Android environment.

Help developers to code highly functional applications-The Android framework may look like "over-engineered" sometimes. Actually, I believe this is a direct consequence of the fact you can do almost everything you want. Unfortunately, this openness (again!) makes it harder to apprehend. Let's say for instance, you are a beginner and wants to develop your own application. You'll have to read a lot of documentation in order to be "up and ready". GreenDroid makes development a lot easier without decreasing the powerfulness of the amazing Android framework!

Leverage the power of the Android framework-Developing on the Android platform may be pretty easy if you're not taking care of the resources you're using. Trying to optimize your application is quite hard sometimes and is a very demanding task. GreenDroid has been developed to be as efficient as possible by integrating basic optimizations. Use as much XML as possible. It's not a mystery to anybody. Android UI development is based on amazing techniques. Layouts and views are defined in XML and automatically inflated by the system. Being an "easy-to-read-for-humans" format, XML is very used among Android developers. GreenDroid puts XML in the middle of the library and takes advantage of all amazing possibilities offered by Android XML files.

VII. CONCLUSION

The problem of dark silicon in both desktop and mobile processor become worse due to utilization wall and The utilization wall forces us to change how we build hardware, Conservation cores use dark silicon to attack the utilization wall GreenDroid will demonstrate the benefits of c-cores for mobile application processors. The greendroid prototype will enable the conservation of dark silicon into energy saving and allow increased parallel execution under strict power budgets .The Greendroid uses C-Core to reduce energy consumption versus Android system and C-core use of the selective depipeling technique to reduce the overhead of executing highly irregular code by minimizing register and clock transitions. According to estimate that the prototype will reduce processor energy consumption by 91 percent for the code that C-Core targets and result in an overall saving of 7.4x. These are great results and since utilization wall is exponentially increasing this way of thinking must be considered in every future architecture both desktop and mobile.

REFERENCES

[1]. G. Venkatesh et al., ‘‘Conservation Cores: Reducing the Energy of Mature Computations,’’ Proc. 15th Int’l Conf .Architectural Support for Programming Languages and Operating Systems, AC M Press, 2010, pp. 205-21.
[2]. N.Goulding et al. “Greendroid: A Mobile Application processor for a future of dark silicon” Hot chips 2010.
[3]. Ankit R.Dongre 1 (BE Student) , Prof Ashish V.Saywan ,” International Journal of Innovative and Emerging Research in Engineering ,Volume 3, Issue 4, 2016 .
[4]. C. Ranjitha, R. Lakshmi Devi,” Special Issue of Engineering and Scientific International Journal (ESIJ) ISSN 2394-187(Online).
[5]. Yepang Liu, Chang Xu, S.C. Cheung and Jian Lü “DOI 10.1109/TSE.2014.2323982, IEEE Transactions on Software Engineering.
[6]. GreenDroid Application Processor Will Battle Dark Silicon - IEEE Spectrum
[7]. “Android Activity Lifecycles.” URL: http://developer.android.com/ guide/components/activities.html.
[8]. “AndroidProcessLifecycle.”URL:http://developer.android.com/reference/android/app/Activity.html#ProcessLifecycle.
[9]. “Android Power Management.” URL: http://developer.android.com reference/android/os/PowerManager.html
[10]. Andrew A. Chien. “A General-purpose Architectural Approach to Heterogeneity and Energy Efficiency, Volume 4, 2011, Pages 1987-1996, Proceedings of the International Conference on Computational Science, ICCS 2011.

BIOGRAPHIES:

	Mr. Andemariam Mebrahtu, Currently I am working as Lecturer and HOD in Department of Computer Science, Eritrea Institute of Technology, Asmara, Eritrea. I have sound experience in teaching, academic Administration activities and research in field of Computer Science. My area of interest includes Cloud Computing, Distributed Computing and Big Data Management.
	Mr. Balu Srinivasulu currently I am working as a Lecturer in the Department of Computer Science, Eritrea Institute of Technology, Asmara, Eritrea. I have wide experience of teaching and research in field of Computer Science. I have published a number of international journal papers related to the Computer Science. My areas of research are Wireless Networks, Communication Networks, Big Data and Cloud Computing.