International Journal of Computational Intelligence Research
ISSN 0973-1873 Volume 13, Number 5 (2017), pp. 805-813

http://www.ripublication.com

Advanced Replicated Hardware Architecture for Smartphones and Smart Devices

Rohit Kumar[1*], Rohit Bajaj[1] and Lokesh Pawar[1]
[1]Chandiharh University, Gharuan, Mohali, Punjab, India.
**Corresponding Author*

Abstract

Smartphones are high end mobile communication devices equipped with computational capabilities similar that of a computer. These mobile devices comes equipped with pre installed software stacks like android, windows etc. These software's are noting but operating systems of these smart communication devices. Conventionally a smartphone hosts a single operating system (OS), and all the applications must be written in accordance with the OS being used. But, if we have an energy efficient and powerful multi-core processor then multiple operating systems can be installed on the smartphone and applications belonging to different platforms can be executed efficiently. In addition the processor is re-con**fig**urable and new OS can installed on a given core as the need arises. In this paper a similar processor architecture has been proposed and implemented and energy consumption has been measured and compared.

I. INTRODUCTION:

Ever since the development of smartphones, these devices has incorporated more and more functions. One of the big problems is that more features mean more chips and more processing cycles, which means higher power consumption. Because batteries do not evolve at the same speed as the appetite of the manufacturers (and buyers) for new features, so a tradeoff always exists between battery and mobile device [1,2].

A typical mobile phone comes with LI-Ion 860 mAh battery which offers approximately 3 watts of energy to perform all its functions until the next refill. A laptop will last for only 5 minutes with this much energy, so energy available for use is major concern. To accurately calculate the total energy stored by the battery, multiply the voltage in volts and the amperage in mAh. For example a battery of

850mAh and 3.7V stores a total of 3,219 mill watts, or 3.219 watts which is very small. A laptop is charged to work for few hours as compared to mobile phone which are supposed to work for days. So it's a tough battle for the designers to achieve the desired motto [3].

On personal computers PC x86 based processors e.g. core2Duo, Phenom, are used for computation. They are highly optimized for performance and comes with very large transistor count. The L1, L2 caches in these systems are very fast and have more capacities for storage, in addition to this they have dedicated units for decoding instruction, scheduling, branch-prediction circuits and multiple execution units per core. To get an idea, a Core 2 Duo E8200 Penryn-based core (which is a relatively small chip by today's standards) has no less than 410 million transistors and has a typical consumption of 65 watts [4,5].

Maker of smartphones are interested in using such capabilities but the heating constraints and power consumption constraints would not let them go e.g. to run a Core 2 Duo processor will require 80 mm copper heat sink and a 6-cell battery which is impossible to achieve with present technology.

That is why no smartphone has been made based on x86 processors architecture. Even low-power processors like the Atom, have an electrical load that is too high for a smartphone and the phone battery would last only for 5 to 10 minutes. The restrictions regarding the size and consumption has made the hardware of smartphones evolve in a way quite different from the PC, using low-power processors and highly integrated chips.

Advanced RISC (Reduced Instruction Set Computer) machines are being used to mitigate the above mentioned problem of x86 processors. ARM (Advanced RISC Machines) are 32-bit RISC processors, with highly optimized architecture and low numbers of transistors and have very low power consumption.

ARM processors are produced in larger volumes and brutally used in all sorts of devices, routers and Asymmetric Digital Subscriber Line (ADSL) modems to video games. Virtually every electronic device you have at home uses a 32 bit ARM processor including smartphones and only exception is your PC [6,7,8].

II. PROPOSED ARCHITECTURE

Figure 1.1 presents the proposed architecture. Each of the major block and components in the proposed architecture has been explained in detail in rest of the chapter.

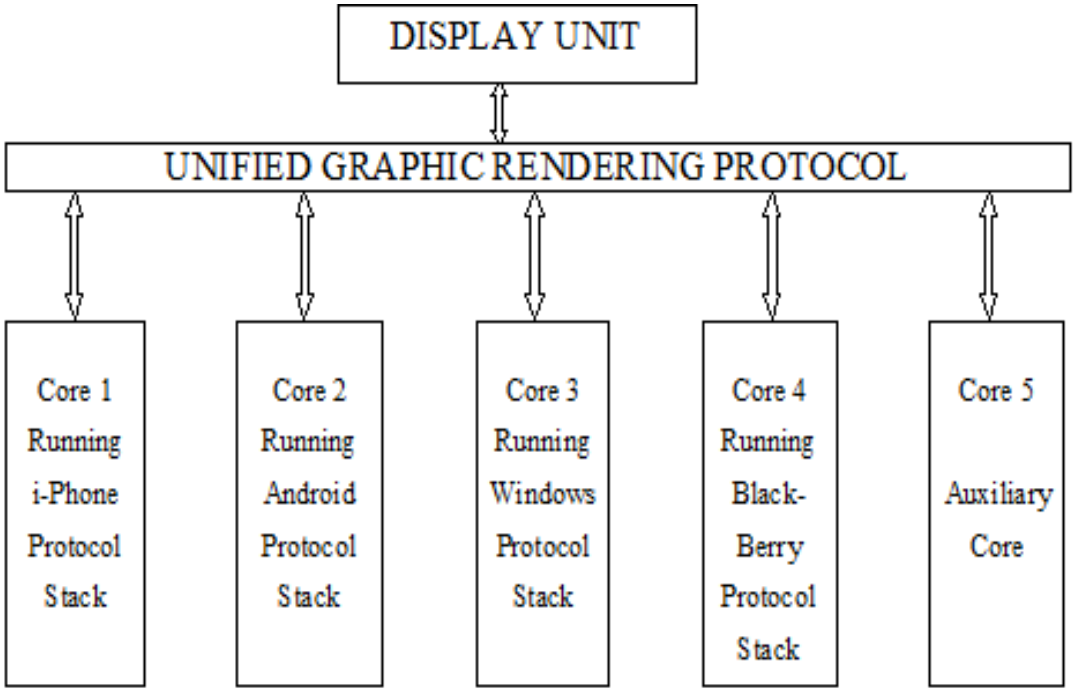

Figure 1.1.: Proposed Architecture

.

2.1 Cores/Processors and Corresponding Protocol Stack

Each of the major core is an independent multi-core processor with many number of energy aware conservative cores. Each of the cores will run a Specific Mobile OS. e.g. the first core is running Android Protocol Stack, other cores are running Symbian, Blackberry, Windows and respectively and one core reserved for future use. Each core is tailored according to the OS to be used. The conservative cores inside each core can configured for any specific code. Conservative cores are used to run some specialized piece of code and the code that also runs frequently. Also, the cores can be reconfigured as the code changes. Each core/processor has conservative cores of various sizes to best fit the code of various sizes and complexity.

2.2 Unified Graphic Rendering Protocol

Each of the mobile platform has its own graphic rendering mechanism, so one need to have a common graphic rendering mechanism that will adapt according to the mobile platform used (i.e. the core being used or when multiple cores are being used).

Since multiple core executes multiple software stacks and each software stack has its own graphic rendering mechanism, so each must be implemented carefully and specifically. With this specialized hardware is used for implementing graphic rendering and graphic library functions for all platforms. This increases the complexity of the problem. In the presented model situation is more grave as multiple

graphic rendering mechanisms need to be implemented and each needs its own specialized hardware as well. Actual implementation of graphic rendering mechanism is outside the scope of this thesis.

2.3 Architecture of Single Core

Figure 1.2 describes the detailed architecture of each core used in the proposed model (Figure 1.1). Each core consists of array of tiles. Each tile of the processor is a full-fledged processor with a set of conservative cores. Number of conservative core in a processor can vary from 8 to 90. Each tile implements code specific to one or set of applications which are closely related and each conservative core within each tile is supposed to run some specialized code from the concerned applications. Tiles are chosen according to the application type and requirement. If an application has many modules then tile with large number of cores will be preferable and will be chosen automatically by the application [9].

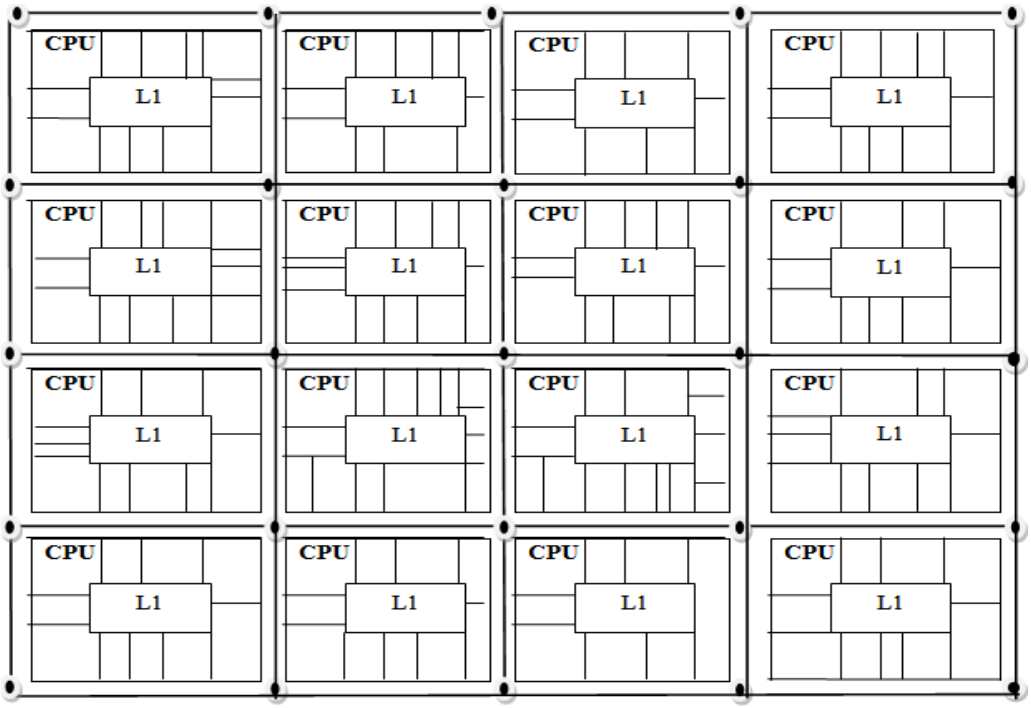

Figure 1.2.: Array of 16 processing units.

Figure 1.3 shows components of a single tile and each of these tile is a full fledged processor in itself. Each tile has on chip network (OCN), instruction cache, data cache and number of conservative cores. Size of cores and number of cores will vary from tile to tile. All conservative cores are tightly connected with the data cache for communication.

C-cores featured here are size variant and comes for different sized codes. The data cache or L1 is tightly integrated with the cores and provides very tight integration among them. All sort of data communication is done through this data cache. Since a single core has sixteen different tiles in a core and each core also have sixteen different L1 data cache and all of these caches must be kept consistent with each other. To bring consistency directory based cache coherence protocols have been used.

When the full system executes only few cores are active in execution at any moment of time so the problem of utilization wall is contained. Due to specialized nature of conservative cores these c-cores consumes very little energy which has been utilized in the presented model.

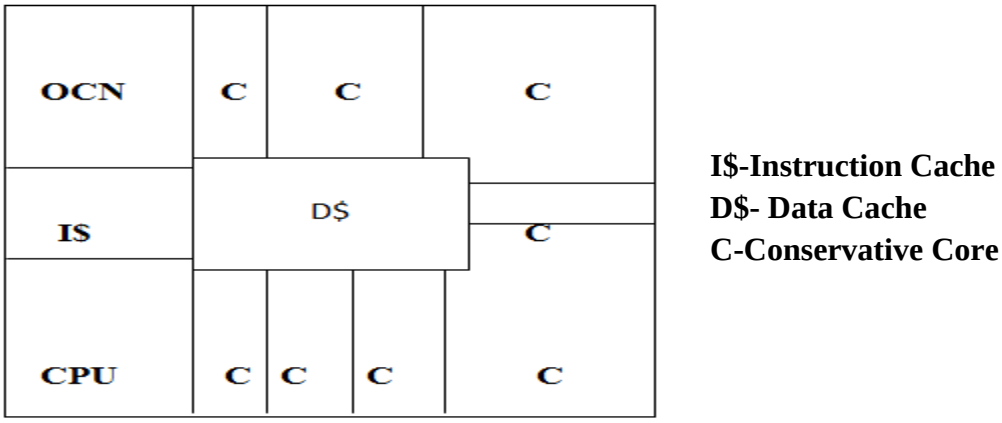

Figure 1.3.: Components of each Tile.

2.4 On Chip Network

The architecture has on chip network which connects all the tiles in the system. Mesh topology is used to connect each tile. Data transfer and synchronization are done by OCN. Since OCN is integrated within the chip the data transfer is very fast.

2.5 Cache Memory

Each tile has dedicated data cache memory with capacity of 32 K-Bytes and is called L1. Each c-core is tightly coupled with L1 cache. All sort of communication is done through these caches like, parameters passing, context switching etc. Besides this instruction cache of 16 Kbytes is embedded in each tile to hold the program instructions.

2.6 Translation Look Aside Buffer

A translation look-aside buffer (TLB) is a cache memory management hardware uses to improve virtual address translation speed. All current desktop, notebook,

smartphones and server processors use a TLB to map virtual and physical address spaces, and it is nearly always present in any hardware which utilizes virtual memory. Each tile in the architecture has its dedicated TLB.

The TLB is typically implemented as content-addressable memory (CAM). The CAM search key is the virtual address and the search result is a physical address. If the requested address is present in the TLB, the CAM search yields a match quickly and the retrieved physical address can be used to access memory. This is called a TLB hit. If the requested address is not in the TLB, it is a miss, and the translation proceeds by looking up the page table in a process called a page walk. The page walk is an expensive process, as it involves reading the contents of multiple memory locations and using them to compute the physical address. After the physical address is determined by the page walk, the virtual address to physical address mapping is entered into the TLB.

Figure 1.4 presents the detailed architecture of each tile. It has been observed during trials that c-cores consume 18 times less energy on average than conventional processor. C-cores comprise most of the execution area so only very small code executes on general CPU. Power gating and clock reduction techniques are used to minimize the power consumption. Power gating disconnects the cores those are not used in present computation to save lots of energy. Clock reduction is used to reduce power consumption when only is being used for computation. When large code is executed clock frequency is increased for efficient computations.

2.7 Cache Coherence

Each tile or node shares its cache memory with other tiles this leads to the problem of cache coherence. Cache coherence implies that each node or tile must be given updated copy of the data modified in other cache. If an old value will be given to the requesting node then it will result in inconsistent computation and thus incorrect information will be represented by the system. In the present model directory based cache coherence protocol has been used. In this protocol each node maintains a directory of the items under its cache memory in the form of bit vectors. If a request comes and node contains updated copy of the item then it responds with the data item. But in case if the data item has been modified by any other node than it forwards the request to that node and modifies its own cache as well.

2.8 Conservative Cores

As already mentioned each tile has conservative cores of various sizes and each chip contains 8 to 90 conservative cores. Idea behind keeping different sized core is to use them for different sized code modules. Small code will be placed on small cores and big cores will be used for larger code.

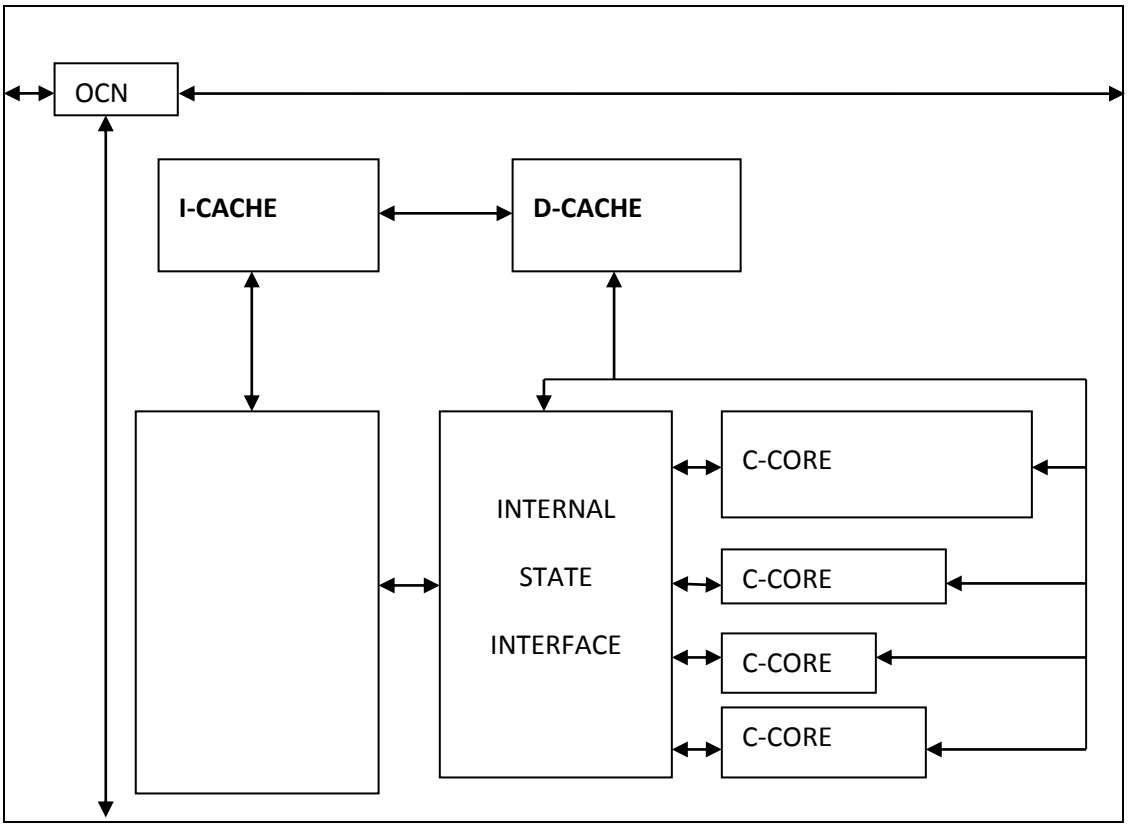

Figure 1.4.: Architecture of Each Tile.

Automated tool exists that finds out the appropriate cores for different application. These cores can also be reconfigured to accommodate code changes and new applications can be installed on them with ease.

2.9 Fine Grained Conservative Cores

When many small sized c-cores exist in the system then the system is said to be a fine grained CPU and has been used in the proposed architecture. By the use of fine grained c-cores majority of the system code can be synthesized in c-cores and during execution c-cores span up to 90 % of execution time. Use of extremely large number of c-cores is not recommended and is not possible either as they increases the communication load on the processor and performance degradation occurs. This problem is also known as multi-processor scaling problem. Due to this the processor chip area must not be divided into very large number of c-cores.

III. RESULTS AND DISCUSSIONS

Energy Savings Achieved by the Use of C-Cores

It is the use of conservative cores that brings lots of energy savings. Conservative cores contains hardwired synthesized version of the software code. Due to hardwired implementation of the code, fetch-decode cycle for executing instruction is eliminated, data path communication overhead is also eliminated and there is no requirement of register file and instruction cache at all, as all the operands and data are directly embedded into hardwired logic except data supplied by user while executing an application.

Table 1.1 presents the energy savings obtained by the use of conservative cores. Conservative cores just need to interact with data cache which amounts to only six percent of the total energy consumed by the baseline CPU. In contrary to it conventional baseline CPU consumes lots of energy in fetch-decode cycle which amounts to 19 % of energy consumption, maximum energy is wasted in data path communication which consumes 38 % of the energy, with this instruction cache consumes lots of energy amounting to 23 %, followed by register file with 14 % percent energy consumption.

In conservative cores energy is consumed by data cache only, rest of the factors has been eliminated in conservative cores which results in very high energy savings and these energy savings needs to be capitalized to make a very efficient processor design. In this thesis work these saving have been capitalized to make the proposed processor prototype and to achieve the desired objectives.

Table 1.1.: Energy Consumption Comparison in Baseline CPU and C-cores.

Factors	Energy consumed in Baseline CPU (%)	Energy Consumed in C-Core (%)
D-Cache	6	6
I-Cache	23	NIL
Fetch/Decode	19	NIL
Register File	14	NIL
Data Path	38	NIL

IV. CONCLUSION AND FUTURE SCOPE

Design of integrated circuits and processors have evolved a lot in the past few decades. Each new processor has eliminated some of the anomalies in the previous version. So there always exists a scope of improvements. In the proposed system work has been carried out by the means of simulation. Though, all types of

applications can be executed on the proposed architecture, but minor re-configurations needs to be done when new protocol stack is to be installed on the proposed architecture or when code change occur. This re-configurability involves programming c-cores to achieve effective and fast execution and to save energy.

Effective and automated tools must be designed which can remove all user botheration when code change occurs. These tools must be highly efficient to detect hot code and its real time synthesis in c-cores. Till now no automated tools exist that removes all user botherations completely and lots of work needs be done to improve these automated tools.

REFERENCES

[1] Mombert, G., http://www.digitaltrends.com/mobile/what-is-asmartphone/ [online], last *seen* dec, 2014.

[2] Johnny John and Chris Riddle, "Smartphone Power*", proceedings of DAC, Anaheim, California, USA*, pp. 935-936, 2010.

[3] Vinay Mehta, http://berylsystems.com/smartphone.pdf [online], *seen* oct, 2010.

[4] Steven Cavanagh and Yingxu Wang, "*Design* of a Real-Time Virtual Machine (RTVM)", *proceedings of Electrical and Computer Engineering 2005 Canadian Conference,* pp. 2021-2024, May, 2005.

[5] Omar A. Fres and Ignacio G. Alonso, "Rovim: A Generic and Extensible Virtual Machine for Mobile Robots", *Fifth International Conference on Systems held in USA*, pp. 37-40, 2010.

[6] Michael Bedford Taylor, "The Raw Microprocessor: A Computational Fabric For Software Circuits And General-Purpose Programs", *IEEE proceedings*, pp. 24-35, 2002.

[7] Robert H. Dennard, "Design of lon-Implanted MOSFET's with Very Small Physical Dimension*", IEEE Journal of Solid-State Circuits*, vol. SC-9, pp. 256-268, October 1974.

[8] Asaf Ashkenazia, Dimitry Akselrodb and Yossi Amona, "Platform Independent Overall Security Architecture in Multi-Processor System-on-Chip ICs for Use in Mobile Phones and Handheld Devices", *proceedings of World Automation Congress (WAC*), vol.33, issue no.5-6, pp. 407-424, 2007.

[9] Meira Levy, Peretz Shoval, Bracha Shapira, Aviram Dayan and Meytal Tubi, "Task Modeling Infrastructure for Analyzing Smartphone Usage" *Ninth International Conference on Mobile Business / 2010 Ninth Global Mobility Roundtable*, pp.264-271, 2010.