Energy and Run Time Evaluation for Compiler Methods Using Dynamic Voltage-Frequency Scaling on XScale Architecture

Khushboo C [#1] and M. Rajasekhara Babu [#2]
SCSE, VIT University
Vellore Institute of Technology, Vellore, India
[1] khushboo.c23@gmail.com
[2] mrajasekharababu@vit.ac.in

***Abstract*— Energy Consumption is a key constraint in modern computing era. The consumption of energy should be ascertainable not only to Gate Level or Register Transfer (RT) Level but also to the System Level. There is a trade-off between energy consumption and run-time. Researchers are working on low power consumption techniques without degrading performance of the system. The computing science developers focus on both compiler and non-compiler methods for energy minimization. This paper focuses on various techniques, methods and tools for minimization of energy without increasing runtime. The energy consumption and run time computed for various compiler techniques on XScale Architecture using XEEMU tool. The best compiler method picked out and code is tuned dynamically by varying voltage-frequency.**

Keyword- Compiler Optimization, Power dissipation, Power Performance, Voltage- Frequency Scaling, XScale Architecture

I. INTRODUCTION

Energy consumption is the major problem for computer systems. With the increasing demand of advanced technology and its utilization in our daily life, it has brought a radical change in our life style. From computer to smart phones, in order to run these devices all we need is power. Scientists and researchers of different fields are working on various techniques to minimize the power consumption. They are trying to reduce the consumption of power on Chip-Level [51], Gate-Level [52], Operating System Level [53], Processors [54] and Compiler Level [55]. Power reduction of computers has been explored from many years. When it comes to computer scientists a steady progress has been achieved basically in the form of Dynamic power management (DPM) and Dynamic voltage scaling (DVS). DPM works on system-level. It minimizes the consumption of power when any parts of the system are not in use. The power manager (PM) is responsible for minimizing power consumption. PM brings the devices to sleep or shut down or wakeup states if it is not in use [2]. On the other hand when system is in running state, to minimize the power consumption DVS plays the role [1]. DVS is used for the processor and as the name says it is real time technique. It reduces the power by varying the voltages according to the load on the processors. Now the question comes why it is important to work on power reduction. Power reduction is required in order to minimize the cooling cost which includes data centres [56] and chip packaging [57], improve battery life time of mobile devices [58], enhance system reliability, above all environmental concerns. Whenever we say power reduction in the field of computing science, processors draw attention the most. Basically for processors it can be obtained in two ways, one by using compiler methods and assembly code manipulation and another by non-compiler methods. The non compiler method checks the load on the processors and dynamically increases or decreases the processor frequency.

In this paper we have discussed various areas where power management can be obtained in the field of computing science along with the methodology of performing energy consumption and run time evaluation. The reminder of the paper is organized as follows: The immediate next section presents the concepts and related literature study. The next section is followed by the methodology, immediate to this some related results and analysis. The paper ends with future enhancement and conclusion.

II. CONCEPTS AND LITERATURE STUDY

Energy performance evaluation is now a day's important research topic as we realized the importance of power. In earlier days systems were not good enough in terms of conserving power. It was using either 100% of their power requirement or was switched off. The era of personal computer power management started in 1989 [59]. Intel introduced the technology which allows CPU to slow down, suspend or shut down [3]. The demand of hand held devices also increased in early 90's. To minimize power consumption scientists started paying attention on sequential improvement of hardware and software both. It has combined to increase the performance and on-battery lifetime of hand held devices [4].In terms of development of personal computers

power management there are three specifications power management, System design and Application. These are 3-interlocked tracks that helped in the development [5]. In terms of computing systems the power and performance management mainly highlights computer architecture, operating systems, CAD, compilers etc. Here, the description starts from operating system first. Operating systems have basically two major roles, providing an abstraction and managing resources like CPU time, memory allocation, disk quota etc. On OS level an efficient power performance can be enhanced by Power Management on IO Devices, Low-Power Scheduling for Processors or by Reducing Memory Power [6]. Energy efficient design also focuses on computation, communication and storage units [5, 6]. In operating system level the concept of hibernates, sleep, auto shutdown etc has also been introduced. The system gets completely or partially powered off, when it is not in use. It allows the system to save power [7]. In other hand the processors are also one of the most important elements of computer systems. When system is in idle state we can save power by keeping focus on operating systems but when systems are in running state it is better to focus on processors [6, 7]. CPU needs more number of cards or PCB i.e. printed circuit boards for large machine, whereas for small workstations & personal computers it holds in one single-chip known as microprocessors. The trend of microprocessors started from 1970's [8]. . In year 2001, after the execution of series of projects named POWER1, POWER2, POWER3, IBM introduced POWER4 processor. It was the first GIGA-series processors. In year 2004 POWER 5 has been introduced as a dual-core processor which supported two-threads simultaneous multithreading, an evolutionary achievement indeed [9, 10]. POWER 6 in year 2007 and POWER 7 in February 2010 came in the picture. POWER 7 holds 8-cores and 4-threads in each core. Hence POWER7 holds the overall capacity of 32 simultaneous threads at a time [11]. Similarly if we talk about Intel from 404 processor series in the year 1972 [12] to Atom last revised in December 2011 [13]. There have been consistent progresses among processors/microprocessors in terms of power and performance evaluation. Because of the evolutionary work after the introduction of microprocessors, The National Academy of Engineering enlisted Microprocessor as one of ten outstanding engineering achievements for the advancement of human welfare in year 1989 [12]. In year 1995 Intel Pentium Pro Processor came in the picture designed for 32-bit server etc. Intel Pentium II Xeon processors in 1998, Followed by Itanium processor which became first in the family of 64-bit products. With the increasing demand and requirement of high performance computer in year 2003 Intel Centrino mobile technology brought a radical changes in terms of high performance, enhanced battery life and lighter PCs. Intel has introduced Dual-core technology and launched four processors for servers under the Xeon 5300 brand. These quad-core processors show improved performance, in year 2006. With the sequential execution of projects on processors, in year 2008 Intel introduced Atom. It is an immediate successor of the Intel A100 & A110 low-power microprocessors. The revised Version of Atom has been introduced in December 2011, which is highly efficient and consumes less power unlike the previous one [14, 15]. Along with IBM and Intel there are other industrial and non-industrial research labs working on power and performance evaluation. The power management for microprocessors can be done over the whole processor, or in specific areas. With dynamic voltage scaling and dynamic frequency scaling, the CPU core voltage, clock rate or both, can be altered to decrease power consumption. Already mentioned in section I the power reduction of processor can be approached in two ways compiler and non compiler. In order to bring power awareness in developer compiler optimization techniques are one of the most approachable ways. In relation with compiler optimization, Proebsting's Law was proposed by Todd Proebsting in the year 1998. According to this law advances in compiler optimization doubles the program performance in every 18 years [16, 24]. The origin of Proebsting's law is famous Moore's Law. Techniques used for optimization can influence from a single statement to the entire program. There are various compiler optimization techniques available. Most general techniques are locally scoped and globally scoped. Few of the scoped optimization techniques are Local [16, 17], Global [18], Inter procedural [19], Machine code [20], whole-program [21], Peephole [22, 23] and Loop optimization [24]. Besides these there are few programming language dependent and machine dependent techniques [23]. Among all types of optimizations the most important technique we say is loop optimization as most of the programs spend large amount of time inside the loop [23, 24]. However, optimizing compilers are by no means perfect. There are so many practical issues and dependencies with optimizing compiler technology are available.

In last decade few remarkable works has been carried away in the field of compiler optimization. In comparison with traditional compilers and optimization techniques, modern compilers provide more number of optimizations. The effect of compiler optimization is highly dependent on the code being compiled. All the optimization techniques have a different impact on code quality, compilation time, code size, energy consumption, etc. That is why compiler provide different optimization level -Os, -O1, -O2 and -O3. It integrates various optimizations. By providing number of exchange between various functions like compilation time, code size & quality etc. Hence a concept called COLE - Compiler Optimization Level Exploration was introduced in the year 2010 [25]. On the other side when we talk about processors, the embedded processors performance is dependent on both underlying architecture and the compiler optimizations applied. To obtained performance of an optimizing compiler on a large micro architectural design. A machine-learning model helps. It predicts the performance of an optimizing compiler without tuning it [26]. Lots of work has been carried out over machine

learning approach [27]. GCC is a compiler which holds good and even called as self-tuning compiler [28, 30]. There are so many practical limitations in optimization. Sometimes the tasks are performed on the same environment (Operating systems, Architecture, Compilers) to check various optimizations. To come over from this 'Collective Optimization' a method has been approached [29]. After the discussion for compiler optimization the discussion for frequency and voltage scaling must be followed. Dynamic Voltage Scaling is one of the important techniques to reduce power consumption. Most importantly the supply voltage and frequency can be adjusted at run time [1]. DVS/DFS finds the program section where voltage and frequency can be tuned on CPU with minimum loss in performance [1, 31]. It is an effective algorithmic technique for minimizing power consumption of embedded real time systems. Unlike the existing DVS a new techniques has been proposed which includes slack estimation. As DVS holds large number of dependency on the slack methods. With the experimental studies it says energy consumption reduced by 20 to 40 percent of current DVS [33]. So many researches have been conducted in the area of dynamic frequency voltage scaling. Maximum implementations are tested on simulation environment and most of the studies highlighted on the minimization of power consumption [34]. But power dissipation is one important issue which was getting overlooked by the scientists. A Baseline measurement has been obtained for the power driven software. The hardware used remained unmodified [35]. In year after 2000 till date so many algorithms have been introduced which covers the flaws of DVS algorithm. Few are new and few are hybrid. The algorithms named dynamic voltage loop scheduling i.e. DVLS is one among them, in year 2007. DVLS is designed by integrating DVS and Loop-scheduling techniques. The experimental results show that DVLS achieves big energy saving [36]. In 2008, DVS algorithm along with linear programming processing has been obtained for real-time multimedia. It has highlighted the power consumption minimization in the field of multi-media [37]. Again a new DVFS technique has been introduced. Usually when systems are in standby mode the energy consumption measurement gets overlooked. This new DVFS has not only measured for active power but also for standby power [38]. In March 2010, combination of DVS and ABB (Adapting Body Biasing) has been introduced. It highlighted the way of minimization of leakage along with power consumption [39]. For many core processors the traditional dynamic voltage and frequency scaling methods treated all the cores identical, in 2011. With new techniques the power of work parallel work load can be obtained independently by varying voltage and frequency. It is named as Dynamic Voltage Frequency Core-count Scaling (DVFCS). The concept of MVMF, application of independent heterogeneous voltage frequency control has been obtained [40]. Instead of calculating power consumption for fixed update interval, an adaptive method has been introduced to perform dynamic voltage and frequency scheduling (DVFS) [39, 40].

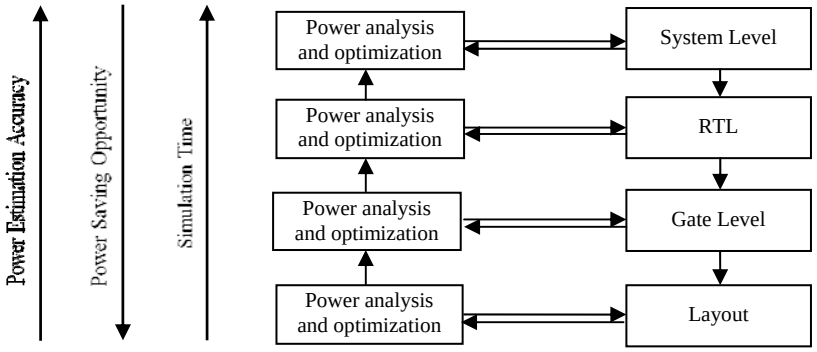

Fig 1. Power analysis and optimization at different levels of the design.

With the discussion of the work carried away till date we came across various factors which play major role in power and performance evaluation, across various domains and even in the field of computer science respectively. In the next sections we will discuss about the various types of power models available followed by various tools available for power performance evaluation. The power models are classified on the basis of the level of abstraction of the description of the system and are reviewed. These are Transistor Level Power Estimation, Gate Level Power Estimation, RT Level Power Estimation, and System Level Power Estimation [32]. The power model gives the measurement of power dissipated or power consumed by any system. According to the abstraction level the effect on power estimation accuracy, simulation time and power saving opportunity is explained in Fig 1. Here Power Estimation Accuracy and Simulation time reducing from bottom to up, whereas Power Saving Opportunity is reducing from top to bottom. Basically two methodologies exist for estimating the power dissipation at different levels of abstraction. These are simulation based methods and probabilistic methods. Along with the power models there are various power analysis tools available now days. It measures the power performance evaluation of laptop to smart phones, from tablet PCs to hand held devices. Scientists are putting great effort in the development of power analysis tools for various devices and its implementation in various domains. The various power analysis tools are Joule Track, WATTCH, Simple Scalar,

XTREM, XEEMU, Simics, Cache Access and Cycle Time Information: CACTI, Simple Power, General Execution-driven Multiprocessor Simulator (GEMS), WARTS - Wisconsin Architectural Research Tool Set. Joule Track is MIT research lab product and a very efficient web based tool for software profiling [41]. WATTCH is CPU power estimation tool. It analyses and optimizes power dissipation at micro architectural level [42], where as Simple Scalar is the complete tool set [45]. XTREM and XEEMU is XScale architecture specific tool [43, 44]. SIMICS is full system simulator [46, 47]. CACTI is the tool for measuring performance based on cache sizes and organization [48]**.** GEMS simulator based on SIMICS [49]. WARTS performs profiling and tracing of the programs [50**]**. Among all XTREM and XEEMU is Intel(c) XScale(c) architecture specific tool. XEEMU developed to simulate the runtime and power consumption of the Intel(c) XScale(c) core. With the experimental results it showed XEEMU is faster and efficient than XTREM [44].

III. METHODOLOGY

After going through the descriptive study related to power performance. Architecture for performing power analysis and evaluation is proposed. It shows the co-relation among the tools and techniques used for minimizing power consumption without affecting the performance. The small description of system, software and the steps required for analysis of energy and power consumption is highlighted in Fig 2.

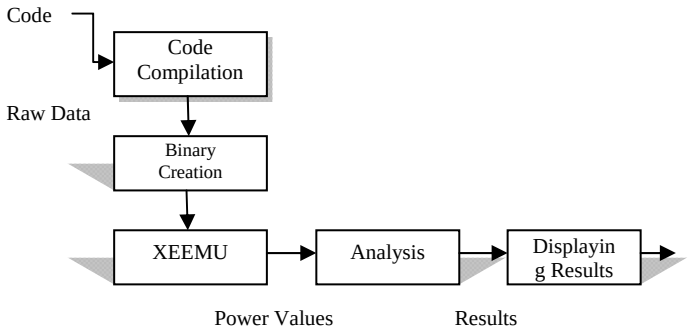

Fig 2. The overview of the steps required for analysis of energy consumption

Here by referring figure 2 after the code compilation the raw data is taken for binary creation. The obtained binary files are taken by the tool to simulate the energy consumption of the program in various power states and also the various compiler optimizations techniques.

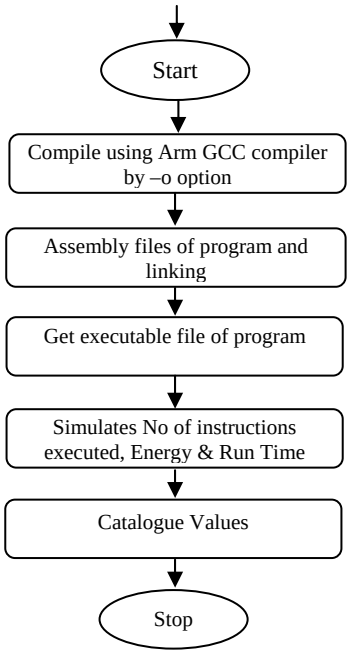

Fig 3. XEEMU Simulation with arm-elf

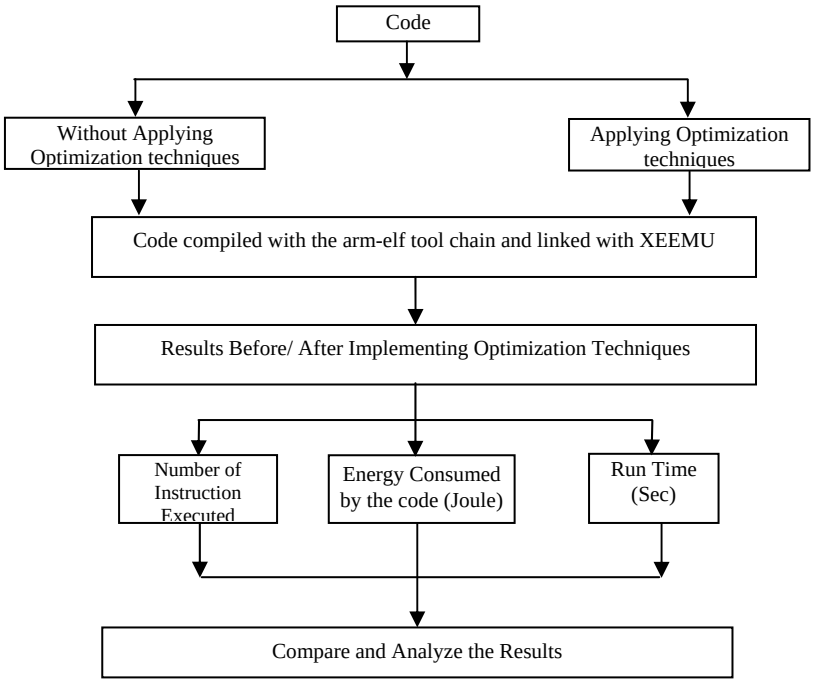

Fig 4. Analyze Results Before and After Optimization

It analyses the obtained values. The results are obtained in the form of number of instructions executed, run time in seconds and energy consumed in joules by the program. The obtained values are analysed and results are displayed. The tool XEEMU used here is a target specific architecture and works precisely for Intel XScale

architecture. So, in order to run the XEEMU linking with arm-elf tool chain is required i.e Xscale Architecture (Intel implementation of ARM). In Fig 3 steps for XEEMU simulation is given. In Fig 4 description of comparative analysis for the code before and after applying optimization techniques is given. The results obtained are compared. The developer can implement different optimization techniques and can choose the one which gives the best result in terms of energy (Joule) and run-time (Sec). The code can be tuned dynamically by varying frequency and voltage across the blocks or the regions in the code. In such a way that minimization in the energy consumption can also be obtained dynamically Fig 5.

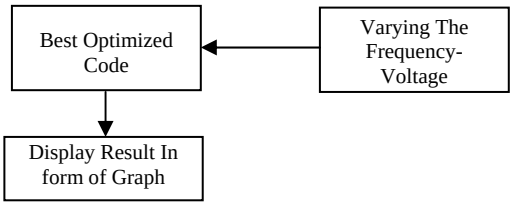

Fig 5. Applying DVFS on the best Optimized Code

IV. FEW RELATED RESULT AND ANALYSIS

The various types of compiler optimization techniques have been implemented. The result gives the changes in the number of instructions executed, energy consumed (Joules) and run time (Sec).

TABLE I

Result Obtained by XEEMU, Before Optimization (OPT_{Before}) and After Optimization (OPT_{After})

Optimization-Techniques		**Number of Instructions Executed**		**Energy Consumed (Joule)**		**Runtime(Secs)**	
	Iterations	**OPT_{Before}**	**OPT_{After}**	**OPT_{Before}**	**OPT_{After}**	**OPT_{Before}**	**OPT_{After}**
Dead Code Elimination	1000	84029672	83529672	0.0953	0.0950	0.2044	0.2037
Loop Termination	1000	83529673	83100671	0.0950	0.2029	0.2037	0.2029
Logic vs Bit	1000	83529673	65544672	0.0950	0.0718	0.2037	0.1544

There is a very apparent achievement in the reduction of energy consumption and minimization of run time after executing the optimized code.In TABLE I three optimization techniques are mentioned for 1000 iterations. Changes in the Values are mentioned. The summation of the energy value and run time has been taken from 500 to 500x10 iterations.

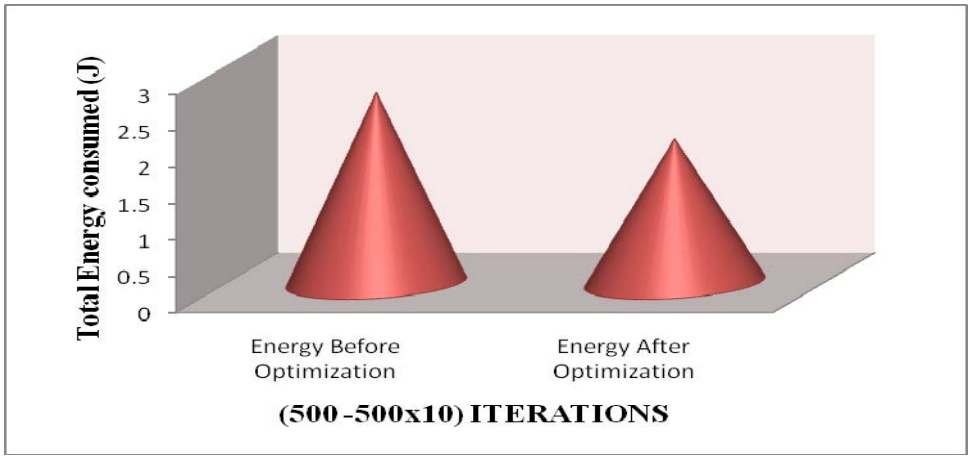

Figure 6: Reduction in energy consumption after applying Logical vs. Bit

The result obtained before optimization compared with the result obtained after optimization, for each three techniques. The total reduction in the consumption of energy and run time for Logical vs Bit is shown in Fig 6 and Fig 7. Among all the three techniques It has given the comparative better result.

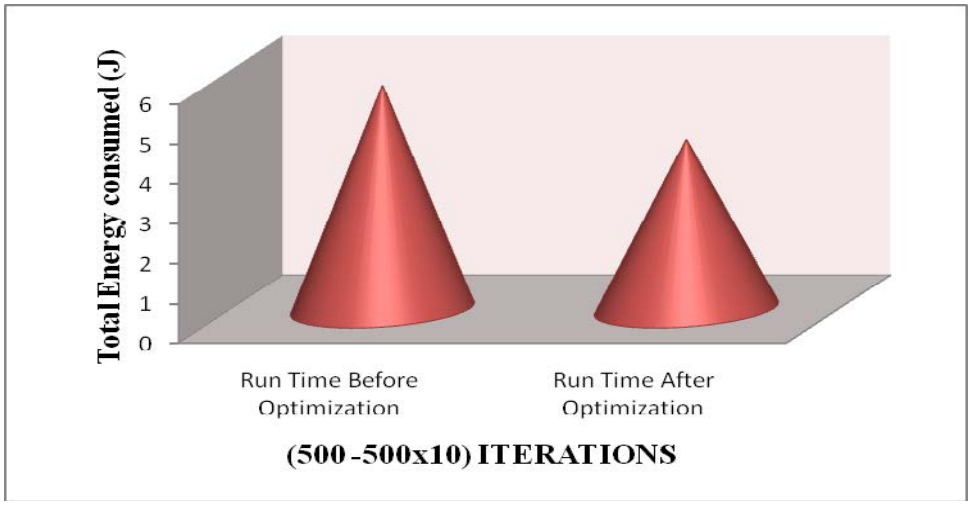

Figure 7: Reduction in run time after applying Logical vs. Bit

V. CONCLUSION AND FUTURE ENHANCEMENT

The less energy consumption and performance evaluation in major areas for the field of computing science has been highlighted. The static and the dynamic both the methods have been taken into the consideration. The performance improvement in terms of energy and run time has been achieved. Here compiler optimization techniques have been implemented on factorial, matrix multiplication; same can be done on very lengthy codes also. The reduction in energy and run time occurs with successive iteration. The different types of voltage-frequency scaling algorithms also can be implemented to tune the code dynamically.

ACKNOWLEDGMENT

We wish to acknowledge XEEMU tool Team (Zoltan Herczeg, Ákos Kiss, Daniel Schmidt, and Balazs Zalanyi) for their consistent feedback regarding the Tool used in the project.

REFERENCES

[1] Zili Shao, Meng Wang, Ying Chen, Chun Xue, Meikang Qiu, Laurence T. Yang, and Edwin H.M. Sha et.al "IEEE Transactions On Circuits And Systems—Ii: Express Briefs, Vol. 54, No. 5, May 2007 , 445 Real-Time Dynamic Voltage Loop Scheduling for Multi-Core Embedded Systems".
[2] http://dynamicpower.sourceforge.net/index.html
[3] http://www.intel.com/intelpress/samples/ppm_chapter.pdf
[4] ACM Transactions on Design Automation of Electronic Systems, Vol. 5, No. 2, April 2000, Pages 115–192 "System-Level Power Optimization: Techniques and Tools" LUCA BENINI Università di Bologna and GIOVANNI DE MICHELI Stanford University.
[5] http://elinux.org/Power_Management_Specification
[6] Yung-Hsiang Lu, Luca Benini, *Member, IEEE*, and Giovanni De Micheli, *Fellow, IEEE* "Power-Aware Operating Systems for Interactive Systems" IEEE Transactions On Very Large Scale Integration (VLSI) Systems, Vol. 10, No. 2, April 2002 119
[7] Xiaotao Liu, Prashant Shenoy and Mark Corner Department of Computer Science, University of Massachusetts Amherst. "Chameleon: Application Level Power Management with Performance Isolation" 2005.
[8] http://www.webopedia.com/TERM/C/CPU.html
[9] https://www.ibm.com/developerworks/power.
[10] IBM POWER Systems Overview [online] https://computing.llnl.gov/tutorials/ibm_sp/
[11] "IBM Journal of Research and Development". Retrieved 2011-07-19.
[12] Chalupsky, Emily Qi. & Ilango Ganga. Intel Corporation "A Brief Tutorial on Power. Management in Computer Systems" David. March 13, 2007.
[13] Anand Lal Shimpi. et.al "Intel's Atom N2600, N2800 & D2700: Cedar Trail, The Heart of the 2012 Net book". Retrieved, 28 December 2011.
[14] http://www.intel.com/content/www/us/en/history/historic-timeline.html
[15] http://www.tayloredge.com/museum/processor/processorhistory.html
[16] Todd Proebsting. Proebsting's Law. Retrieved October 20, 2009 ,http://research.microsoft.com/enus/um/people/toddpro/papers/law.htm, 1998
[17] Larry M. Masinter and L. Peter Deutsch, Xerox Palo.Alto Research Center , ACM proceedings "Local Optimization in a Compiler for Stack-based Lisp Machines" 1980.
[18] John Cooke,IBM Corporation "Global Common Subexpression Elimination", ACM proceedings 1970, page 20-24.
[19] Stephen Richardson , Mahadevan Ganapathi "Interprocedural optimization: Experimental results", published online: 30 OCT 2006 DOI: 10.1002/spe.4380190205.
[20] "Machine Code Optimization" [online] cs.nyu.edu/web/Research/Theses/old/goss_clinton.html.
[21] Spyridon Triantafyllis Matthew J. Bridges Easwaran Raman Guilherme Ottoni David I "A Framework for Unrestricted Whole-ProgramOptimization",August Department of ComputerScience,PrincetonUniversity{strianta,mbridges,eraman,ottoni,august}@princeton.edu. June 10–16, 2006, Ottawa, Canada Copyright c 2006 ACM 1-59593-320-4/06/0006.
[22] Jack W. Davidson And David B. Whalley **"**Quick Compilers Using Peephole Optimization**"** Department of Computer Science, University of Virginia, Charlottesville, VA 22903, U.S.A.

[23] By Alfred V. Aho, Ravi Sethi, Jeffrey D. Ullman, Compilers: Principles, Techniques, and Tools Description, “Compilers Principles Techniques and Tools 2nd Edition”

[24] Dan Richard Kaiser et.al “Loop Optimization Techniques on Multi-Issue Architectures” A Dissertation Submitted in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy (Computer and Communication Sciences), In the University of Michigan 1994.

[25] Kenneth Hoste, Lieven Eeckhout et.al “COLE: Compiler Optimization Level Exploration” Kenneth Hoste Lieven Eeckhout ELIS Department, Ghent University Sint Pietersnieuwstraat 41, B-9000 Gent, Belgium {kehoste,leeckhou}@elis.UGent 2010.

[26] Christophe Dubach, Timothy M. Jones And Michael F.P. O’boyle et.al “Exploring and Predicting the Effects of Microarchitectural Parameters and Compiler Optimisations on Performance and Energy” School of Informatics University of Edinburgh, UK christophe.dubach@ed.ac.uk,{tjones1,mob}@inf.ed.ac.uk. 2005

[27] John Thomson, Michael O’Boyle, Grigori Fursin, Björn Franke, “ Reducing Training Time in a One-Shot Machine Learning-Based Compiler” , Languages and Compilers for Parallel Computing , Lecture Notes in Computer Science Volume 5898, 2010, pp 399-407 , 2010.

[28] Grigori Fursin ; Yuriy Kashnikov ; Abdul Memon ; Zbigniew Chamski ; Olivier Temam ; Mircea Namolaru ; Elad Yom-Tov ; Bilha Mendelson ; Ayal Zaks ; Eric Courtois ; Francois Bodin ; Phil Barnard ; Elton Ashton ; Edwin Bonilla ; John Thomson ; Christopher Williams ; Michael O’Boyle “Milepost GCC: Machine Learning Enabled Self-tuning Compiler” Contribution to journal: Article, 296-327, Volume 39, Year 2011.

[29] ACM Transactions on Architecture and Code Optimization, Vol. 7, No. 4, Article 20, Pub. date: December 2010. “Collective Optimization: A Practical Collaborative Approach” Grigori Fursin And Olivier Temam, Inria Saclay, France.

[30] Grigori Fursin *INRIA Saclay, France HiPEAC member*, “Collective Tuning Initiative: automating and accelerating development and optimization of computing systems” grigori.fursin@inria.fr. 2009

[31] Chung-Hsing Hsu and Ulrich Kremer Department of Computer Science Rutgers, The State University of New Jersey *{*chunghsu,uli*}*@cs.rutgers.edu, PLDI’03, June 9–11, 2003, San Diego, California, USA. Copyright 2003 ACM 1-58113-662-5/03/0006

[32] Fayez Gebali and Haytham El Miligi *University of Victoria, British Columbia*, “The Design, Implementation, and Evaluation of a Compiler Algorithm for CPU Energy Reduction Embedded Multi-Core Systems” *Georgios Kornaros (Ed.) 2010* Series Editors.

[33] Woonseok Kim, Jihong Kim, Sang Lyul Min , “A Dynamic Voltage Scaling Algorithm for Dynamic-Priority Hard Real-Time Systems Using Slack Time Analysis” year 2001.

[34] David Tam tamda@eecg.toronto.edu , Winnie Tsang wtsang@dgp.toronto.edu, Catalin Drula catalin@cs.toronto.edu. “Dynamic Voltage Scaling in Mobile Devices” CSC2228 Project Final Report December 15, 2003

[35] “Baseline Measurement of Software Driven Power Consumption” Gary Cameron ATI Technologies Inc., 33 Commerce Valley Drive East, Markham, Ontario, Canada, E–mail: gcameron@ati.com, gfcameron@gmail.com 0-7803-8879-8/05/$20.00 ©2005 IEEE

[36] Zili Shao, Meng Wang, Ying Chen, Chun Xue, Meikang Qiu, Laurence T. Yang, and Edwin H. -M. Sha, IEEE Transactions On Circuits And Systems—Ii: Express Briefs, Vol. 54, No. 5, May 2007 445 “Real-Time Dynamic Voltage Loop Scheduling for Multi-Core Embedded Systems” “Optimality and Improvement of Dynamic Voltage Scaling Algorithms for Multimedia Applications” , DAC 2008, June 8-13,2008, ACM 978-1-60558-115-6/08/006..5.00 by Zhen Cao, Brian Foo, Lei He and Mihaela van der Schaar, Electronic Engineering Department, UCLA Los Angeles, CA 90095 .

[37] Kihwan Choi, Wonbok Lee, Ramakrishna Soma, and Massoud Pedram Department of EE-Systems, University of Southern California, Los Angeles, CA90089 {kihwanch, wonbokle, rsoma, pedram}@usc.edu “Dynamic Voltage and Frequency Scaling Under a Precise Energy Model Considering Variable and Fixed Components of the System Power Dissipation”.

[38] Meikang Qiu, Laurence T. Yang, Zili Shao, and Edwin H.-M. Sha , “Dynamic and Leakage Energy Minimization With Soft Real-Time Loop Scheduling and Voltage Assignment”, IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS, VOL. 18, NO. 3, MARCH 2010 501.

[39] Saurabh Dighe, *Member, IEEE*, Sriram R. Vangal, *Member, IEEE*, Paolo Aseron, Shasi Kumar, Tiju Jacob, Keith A. Bowman, *Member, IEEE*, Jason Howard, *Member, IEEE*, James Tschanz, *Member, IEEE*, Vasantha Erraguntla, *Member, IEEE*, Nitin Borkar, Vivek K. De, *Senior Member, IEEE*, and Shekhar Borkar, *Fellow, IEEE*. “Within-Die Variation-Aware Dynamic-Voltage-Frequency-Scaling With Optimal Core Allocation and Thread Hopping for the 80-Core TeraFLOPS Processor” , IEEE Journal Of Solid-State Circuits, Vol. 46, No. 1, January 2011

[40] Mostafa E. Salehi, Mehrzad Samadi, Mehrdad Najibi, Ali Afzali-Kusha, Masoud Pedram, and Sied Mehdi Fakhraie. “Dynamic Voltage and Frequency Scheduling for Embedded Processors Considering Power/Performance Tradeoffs”, IEEE Transactions On Very Large Scale Integration (Vlsi) Systems, Vol. 19, No. 10, October 2011.

[41] *Amit Sinha - 2001* “JouleTrack - *A Web Based Tool for Software EnergyProfiling”,[online]*mtlweb.mit.edu/researchgroups/icsystems/pubs/.../sinha_dac01.pdf

[42] Tiwari, V, Martonosi, M. “Computer Architecture, 2000 , Proceedings of the 27th International Symposium on 14-14 June 2000” Brooks,D. Dept. of Electr. Eng., Princeton Univ., NJ, USA

[43] http://www.inf.u-szeged.hu/xeemu/page_what_is.html.

[44] Zolt´an Herczeg1, ´Akos Kiss1, Daniel Schmidt2, NorbertWehn2, and Tibor Gyim´othy1 “XEEMU: An Improved XScale Power Simulator”, PATMOS conference held in Gothenburg, Sweden, in September 2007.

[45] http://www.simplescalar.com/

[46] https://wiki.cc.gatech.edu/multicore/index.php/SIMICS_-_A_full system_extensible_multiprocessor_simulator

[47] http://test.virtutech.com/academia/licensing.html

[48] CACTI 2.0: An Integrated Cache Timing and Power Model Glen Reinman and Norman P. Jouppi, February 2000

[49] Brook’s D. M., Bose, P., Schuster, S. E., Jacobson, H., Kudva., P. N., Buyuktosunoglu, A., Wellman, J-D., Zyuban, V., Gupta, M., and Cook, P.W.2000. et.al “Power Aware Micro architecture: Design and Modeling Challenges for next generation microprocessors”. IEEE Micro 20(6):26-44.

[50] http://pages.cs.wisc.edu/~larus/warts.html

[51] IEEE Transactions On Computer-Aided Design Of Integrated Circuits And Systems, Vol. 17, No. 11, November 1998-1099, Gate-Level Power Estimation Using Tagged Probabilistic Simulation, Chih-Shun Ding, Chi-Ying Tsui, Member, IEEE , and Massoud Pedram, Member, IEEE.

[52] Quality Electronic Design (ISQED), 2011 12th International Symposium, 14-16 March 2011, Zhigang Hao Sch. of Microelectron., Shanghai Jiao Tong Univ., Shanghai, China, Tan, S.X.-D.; Guoyong Shi.

[53] Real Time Power Estimation and Thread Scheduling via Performance Counters, Karan Singh Major Bhadauria Computer Systems Laboratory Cornell University, Ithaca, NY, USA, Sally A. McKee Department of Computer Science and Engineering Chalmers University of Technology Goteborg, Sweden.

[54] Instruction-level power consumption estimation of embedded processors for low-power applications S. Nikolaidis, Th. LaopoulosCorresponding author contact information, E-mail the corresponding author Electronics Laboratory, Physics Department, Aristotle University of Thessaloniki, Thessaloniki, 54006, Greece
[55] "Compiler Optimizations For Low Power Systems" Mahmut Kandemir, N. Vijaykrishnan, Mary Jane Irwin , Microsystems Design Lab, Pennsylvania State University University Park, PA , mckee@chalmers.s
[56] An Energy Aware Framework for Virtual Machine Placement in Cloud Federated Data Centres , 2012, Corentin Dupont, CREATE-NET,Giovanni Giuliani, HP Italy Innovation Centre, Fabien Hermenier, OASIS Team, INRIA–CNRS–I3S, University of Sophia-Antipolis
[57] Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. IEEE International, 26-30 March 2007 Freeh, Vincent W. Dept. of Comput. Sci., North Carolina State Univ., Raleigh, NC Bletsch, T.K.; Rawson, F.L.
[58] Fidelity-Aware Replication for Mobile Devices, Kaushik Veeraraghavan, Venugopalan Ramasubramanian, Microsoft Research Silicon Valley
[59] www.cs.cmu.edu/~fgandon/documents/lecture/uk1999