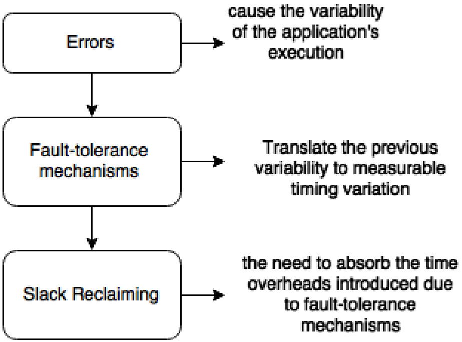
Mitigation of performance variability induced by Checkpoint-Restart using DVFS

Kokolis Apostolos

apostolis_kok@hotmail.com

July 28, 2015

Urgency for slack Reclaiming


Errors
cause the variability of the application's execution
Fault-tolerance mechanisms
Translate the previous variability to measurable timing variation
Slack Reclaiming
the need to absorb the time overheads introduced due to fault-tolerance mechanisms

- Reliability Availability Serviceability (RAS) mechanisms
- Fault-tolerance mechanisms produce the idea of Reliability Wall [42]
- Darpa, Perfect project [16]

Error Profilling

Two types of errors are taken into account [32, 33]:

- Silent Data Corruptions (SDC), where erroneous outputs are generated. Data corruption may not manifest as failure and the application continues its execution
- Detected Unrecoverable Errors (DUE), the application is either terminated or blocked.

Errors are expressed by Mean Time to Failure (MTTF) and Failures in Time (FIT).

Operation of Checkpoint Restart

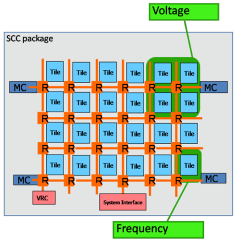
Checkpoint/Restart enables fault-tolerance in a system.

- Stores snapshots of the system or application state (system and application level C/R)
- After failure, operation can be restarted from a previous checkpoint
- C/R can be facilitated in distributed systems through system coordination [41]
- Available tools implementing the C/R technique [12, 31]

Playing with power on CPU

- Sprinting (pushing the TDP) [35, 36]
  - frequency sprinting
  - parallel sprinting
  - both parallel and frequency sprinting
- Boosting (respecting the TDP)
  - Intel Turbo Boost [14]
- DVFS (using the manufacturer defined P-states)

Target Platforms



The Single-Chip Cloud Computer (SCC) [6, 5]:

- Voltage and frequency islands
- 48 homogenuous cores
- `/shared` directory between SCC cores and MCPC

x86 commercial platform:

- Intel(R) Core(TM) i7 2630QM Sandy-bridge
- quad core processor with two simultaneous multi-threading (SMT) contexts per core
- acpi-cpufreq driver

Voltage and Frequency relation

Voltage regulators keep voltages within the desired range

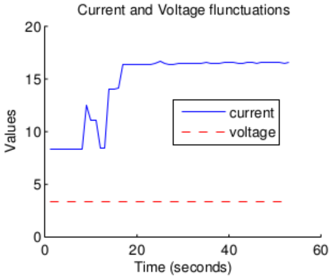
- off-chip
- on-chip

Opperating Performance Point List depict the connection between frequency and voltage

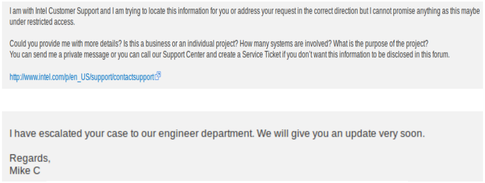
Measuring Energy consumption

For the SCC platform

- we use *sccBmc -c status* to read voltage and current
- we integrate using trapezoidal rule to calculate the Energy consumption



For the x86 platform in discussion with Intel engineers

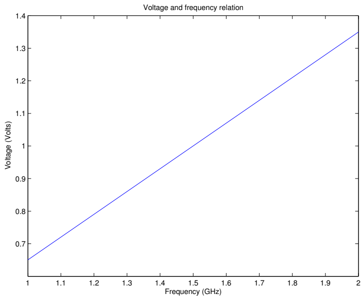

I am with Intel Customer Support and I am trying to locate this information for you or address your request in the correct direction but I cannot promise anything as this maybe under restricted access.

Could you provide me with more details? Is this a business or an individual project? How many systems are involved? What is the purpose of the project?
You can send me a private message or you can call our Support Center and create a Service Ticket if you don't want this information to be disclosed in this forum.

http://www.intel.com/p/en_US/support/contactsupport

I have escalated your case to our engineer department. We will give you an update very soon.

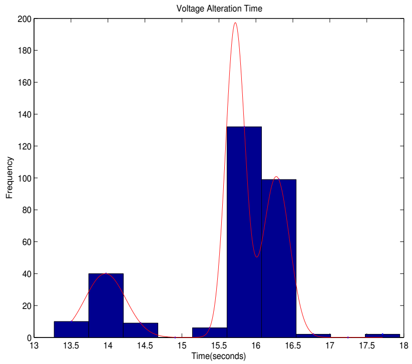
Regards,
Mike C



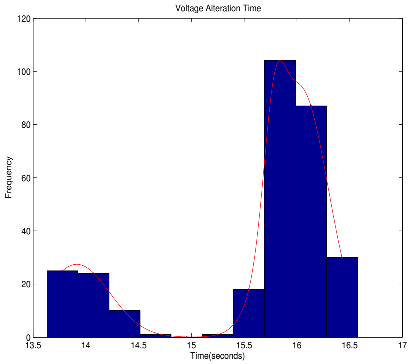
SCC DVFS and overheads

RCCE library provides `RCCE_iset_power()`

From 533 MHz to 800 MHz

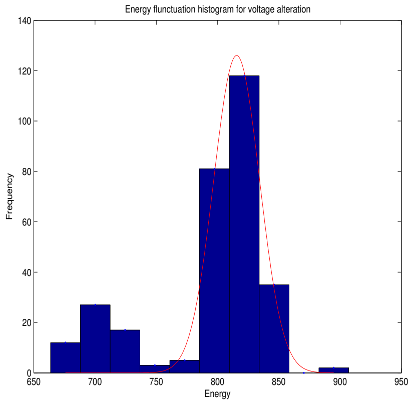


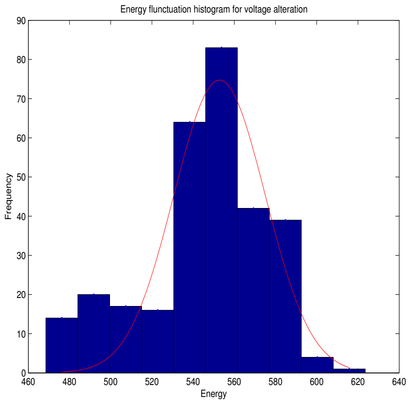
From 800 MHz to 533 MHz



From 533 MHz to 800 MHz

From 800 MHz to 533 MHz





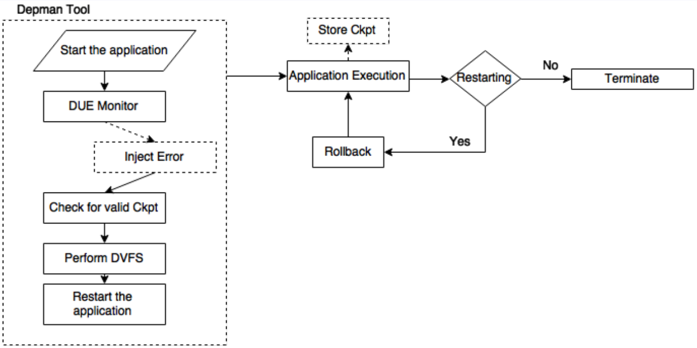
x86 DVFS

CPUfreq governors:

- Performance
- Powersave
- Ondemand
- Conservative
- Userspace, the one used for our scheme

Alterations are performed by changing the `/sys/devices/system/cpu/cpu*/cpufreq` filesystem.
`scaling_setspeed` file is responsible for frequency changes
`cpufrequtils` is used for easier frequency transitions.
Transition latency is about 10 $\mu seconds$

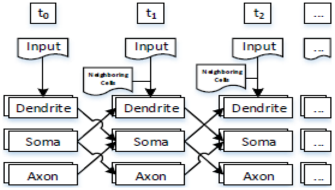
Depman Operation ([31])



Diagnostics: Depman is capable of detecting DUE errors by parsing the application's stdout.
Self Injection: The self injection module induce errors to the execution of the application, with a Weibull distributed probability of

$$P_s = 1 - e^{-\Delta t / MTTF} \tag{1}$$

Target Application and C/R procedure



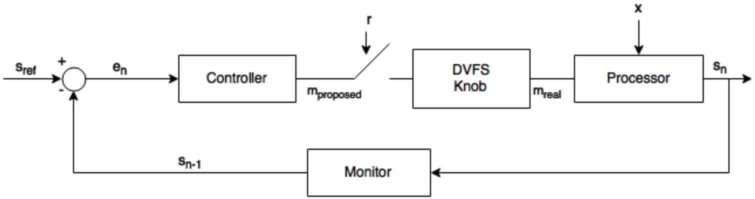
Our target application is the Infoli simulator [15], implementing data parallelism, based on the Hodgkin-Huxley model [24]. We use double buffered checkpoint files to save the application state, which contain:

- simulation step
- the number of cells
- the number of cores
- soma,dentrite,axon compartments

The Restart procedure consists of:

- the restoration of data from checkpoint files
- finding maximum simulation step to restart
- performing output reconstruction if needed

DVFS module



DVFS module work as a Finite-State Machine (FSM). PID control implementations have been proposed [37]

s	application slack
m	frequency multiplier
x	timing noise
r	number of errors per DVFS change
e_n	$s_{\mathtt{ref}} - s_{\mathtt{n-1}}$

Timing Noise Quantification

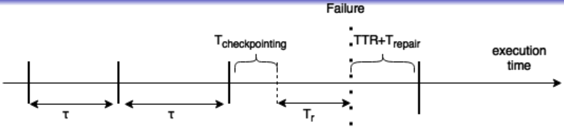
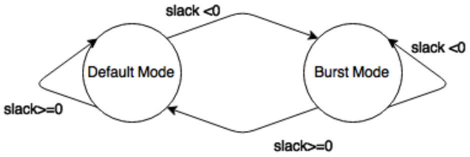


Figure: Modelling the Checkpoint cycle

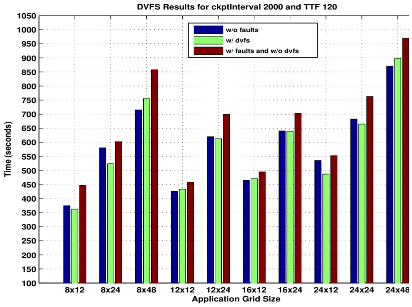
τ is the Checkpoint Interval

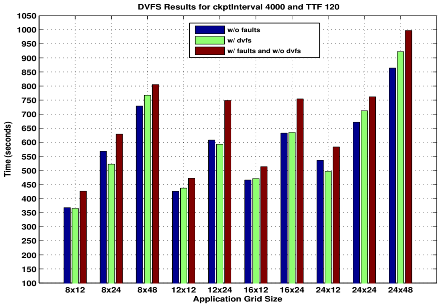
- Rollback Time (T_r)
- Time to Restart (TTR)
- Repair Time (T_{repair})
- Checkpoint Time ($T_{\text{checkpointing}}$)

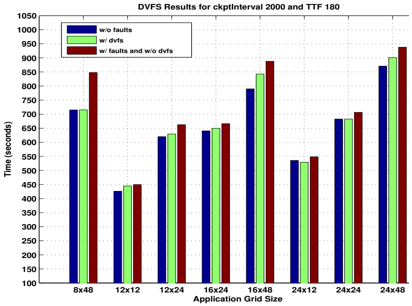
SCC implementation

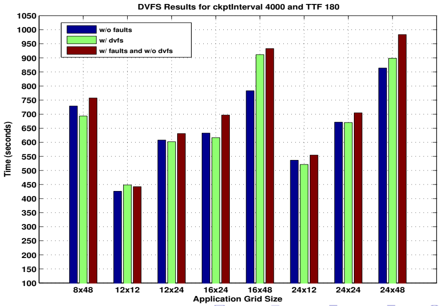


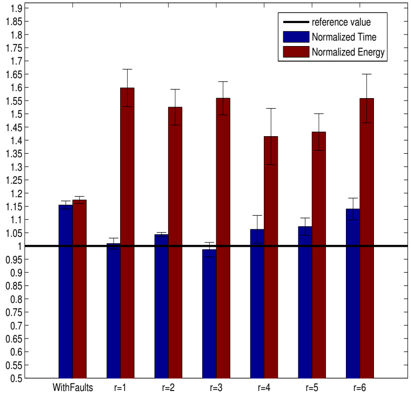
$$\begin{aligned} s_{new} &= s_{previous} - timeOverheads + \\ &+ \underbrace{(lastTTF \times curFreq - lastTTF \times defFreq)}_{\text{depicts the time overhead that has been reclaimed}} \end{aligned} \tag{2}$$

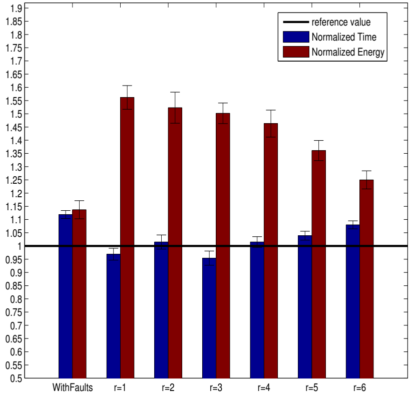

DVFS Results for ckptInterval 2000 and TTF 120
w/o faults
w/ dvfs
w/ faults and w/o dvfs
Time (seconds)
Application Grid Size
8x12
8x24
8x48
12x12
12x24
16x12
16x24
24x12
24x24
24x48


DVFS Results for ckptInterval 4000 and TTF 120
w/o faults
w/ dvfs
w/ faults and w/o dvfs
Time (seconds)
Application Grid Size
8x12
8x24
8x48
12x12
12x24
16x12
16x24
24x12
24x24
24x48


DVFS Results for ckptInterval 2000 and TTF 180
w/o faults
w/ dvfs
w/ faults and w/o dvfs
Time (seconds)
Application Grid Size
8x48
12x12
12x24
16x24
16x48
24x12
24x24
24x48

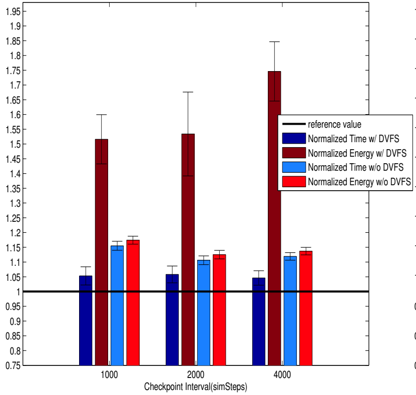

DVFS Results for ckptInterval 4000 and TTF 180
w/o faults
w/ dvfs
w/ faults and w/o dvfs
Time (seconds)
Application Grid Size
8x48
12x12
12x24
16x24
16x48
24x12
24x24
24x48



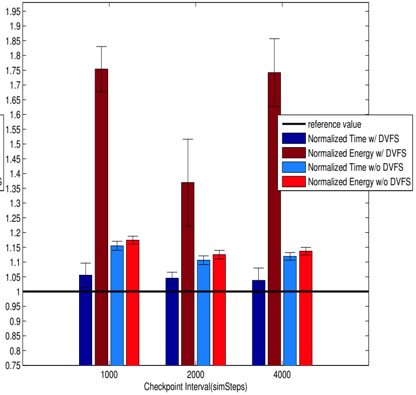


ckptInt=1000 ckptInt=4000

SCC Results, TTF: 122 sec, Time Reference: 823 seconds and Energy Reference: 29210 Joules



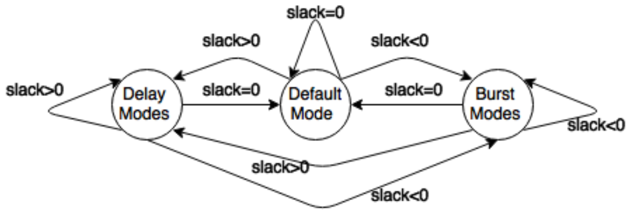
r=1



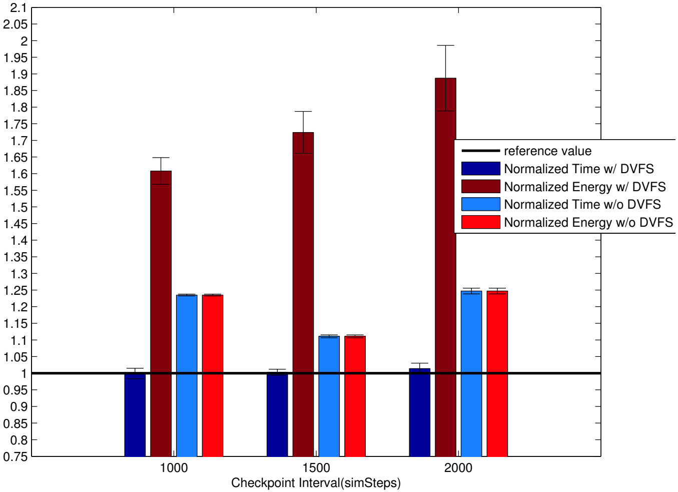
r=3

SCC Results, Weibull distributed TTF, Time Reference: 823 seconds and Energy Reference: 29210 Joules

x86 First Implementation

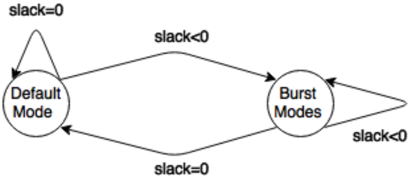


$$newFreq = \frac{-s + defFreq \times MTTF}{MTTF} \tag{3}$$

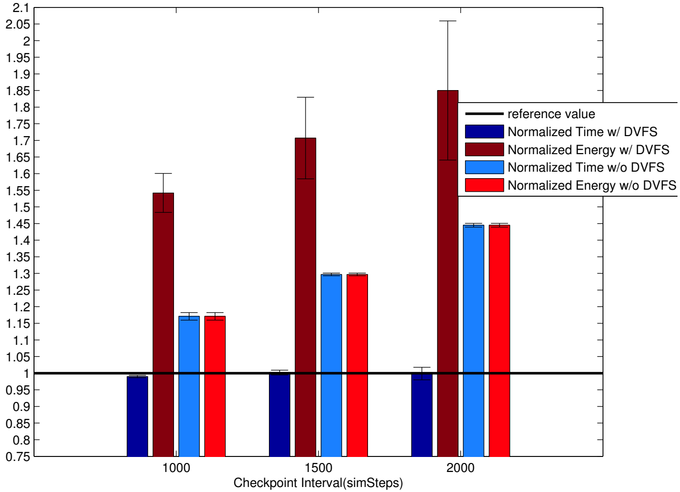


Weibull distributed TTF, Time Reference: 578 seconds and Energy normalized, based on $P \propto f \times V_{dd}^2$

x86 Second Implementation



$$reclaimingTime = \frac{-s}{determinedFreq - defFreq} \tag{4}$$



Weibull distributed TTF, Time Reference: 494 seconds and Energy

Energy normalized, based on P $\propto$ f $\times V_{dd}^2$

Conclusions

- Depman is operating on any platform and can be easily adjusted to any application
- Our scheme needs no additional run time parameters as it can estimate the total time overheads on the fly and perform the available countermeasures
- It seems that the DVFS module is capable of absorbing the time overheads introduced by the C/R procedure
- The additional feature of dependable performance comes at an energy cost

Future Work

- Adopt parallel sprinting besides DVFS
- Use Depman as a main component for distributed systems C/R and synchronization
- Export the DVFS module as a library used by programmers
- Adopt DVFS techniques by web-servers to improve responsiveness
- Modify Depman from the fail-stop model, as to react to multiple events

Acknowledgements

I would like to personally thank Professor D.Soudris and Phd candidate D.Rodopoulos for their guidance and support throughout our cooperation.

Thank you all for your attention!

References I

[1] The linux kernel documentation.
https://www.kernel.org/doc.

[2] Python software foundation. python language reference, version 2.7. available at http://www.python.org.

[3] Texas instruments, dvfs user guide.
http://processors.wiki.ti.com/index.php/DVFS_User_Guide.

[4] Ubuntu manpages (cpufrequtils).
http://manpages.ubuntu.com/manpages/lucid/man1/cpufreq-set.1.html.

[5] *SCC External Architecture Specification (EAS) Revision 1.1.*
Intel Corporation, 2010.

[6] *The SCC Programmer's Guide-Revision 0.75.*
Intel Corporation, Boston,MA,USA, 2010.

[7] 2nd generation intel core tm processor family mobile and intel celeron processor family mobile, September 2012.

[8] J. Ansel, K. Arya, and G. Cooperman.
Dmtcp: Transparent checkpointing for cluster computations and the desktop.
In *Parallel Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on*, pages 1–12, May 2009.

[9] R. Bakker, M.W. Van Tol, and A.D. Pimentel.
Emulating asymmetric mpsocs on the intel scc many-core processor.
In *Parallel, Distributed and Network-Based Processing (PDP), 2014 22nd Euromicro International Conference on*, pages 520–527, Feb 2014.

References II

[10] R.C. Baumann.
Radiation-induced soft errors in advanced semiconductor technologies.
Device and Materials Reliability, IEEE Transactions on, 5(3):305–316, Sept 2005.

[11] Paolo Bazzigaluppi, Jornt R De Gruijl, Ruben S Van Der Giessen, Sara Khosrovani, Chris I De Zeeuw, and Marcel TG De Jeu.
Olivary subthreshold oscillations and burst activity revisited.
Frontiers in neural circuits, (6), 2012.

[12] Greg Bronevetsky, Daniel Marques, Keshav Pingali, and Paul Stodghill.
Automated application-level checkpointing of mpi programs.
ACM Sigplan Notices, 38(10):84–94, 2003.

[13] Yu Cao, J. Velamala, K. Sutaria, M.S.-W. Chen, J. Ahlbin, I. Sanchez Esqueda, M. Bajura, and M. Fritze.
Cross-layer modeling and simulation of circuit reliability.
Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, 33(1):8–23, Jan 2014.

[14] J. Charles, P. Jassi, N.S. Ananth, A. Sadat, and A. Fedorova.
Evaluation of the intel x00ae; core x2122; i7 turbo boost feature.
In *Workload Characterization, 2009. IISWC 2009. IEEE International Symposium on*, pages 188–197, Oct 2009.

[15] George Chatzikonstantis.
Energy aware mapping of a biologically accurate inferior olive cell model on the single-chip cloud computer.
Bachelor thesis, National Technical University of Athens, September 2013.

References III

[16] Dr. Joseph Cross.
Power Efficiency Revolution for Embedded Computing Technologies (perfect).
Technical report, Defense Advanced Research Projects Agency (DARPA), 2010.
Available: `http://www.darpa.mil/program/power-efficiency-revolution-for-embedded-computing-technologies`.

[17] Anand Dixit, Raymond Heald, and Alan Wood.
Trends from ten years of soft error experimentation.
System Effects of Logic Soft Errors (SELSE), 2009.

[18] R.G. Dreslinski, M. Wieckowski, D. Blaauw, D. Sylvester, and T. Mudge.
Near-threshold computing: Reclaiming moore's law through energy efficient integrated circuits.
Proceedings of the IEEE, 98(2):253–266, Feb 2010.

[19] Message P Forum.
Mpi: A message-passing interface standard.
Technical report, Knoxville, TN, USA, 1994.

[20] Juliana Gjanci.
On-Chip Voltage Regulation for Power Management in System-on-Chip.
Master's thesis, B.S. University of Illinois, Chicago, 2006.

[21] Jornt R De Gruijl, Paolo Bazzigaluppi, Marcel TG de Jeu, and Chris I De Zeeuw.
Climbing fiber burst size and olivary sub-threshold oscillations in a network setting.
PLoS computational biology, page 8(12):e1002814, 2012.

[22] D. Hardy, I. Sideris, N. Ladas, and Y. Sazeides.
The performance vulnerability of architectural and non-architectural arrays to permanent faults.
In *Microarchitecture (MICRO), 2012 45th Annual IEEE/ACM International Symposium on*, pages 48–59, Dec 2012.

References IV

[23] Paul H Hargrove and Jason C Duell.
Berkeley lab checkpoint/restart (blcr) for linux clusters.
In *Journal of Physics: Conference Series*, volume 46, page 494. IOP Publishing, 2006.

[24] Alan L Hodgkin and Andrew F Huxley.
A quantitative description of membrane current and its application to conduction and excitation in nerve.
The Journal of physiology, page 117(4):500, 1952.

[25] Jenifer Hopper.
Reduce Linux power consumption.
Technical report, IBM, 09 2009.

[26] Jason Howard, Saurabh Dighe, Sriram R Vangal, Gregory Ruhl, Nitin Borkar, Shailendra Jain, Vasantha Erraguntla, Michael Konow, Michael Riepen, Matthias Gries, et al.
A 48-core ia-32 processor in 45 nm cmos using on-die message-passing and dvfs for performance and power scaling.
Solid-State Circuits, IEEE Journal of, 46(1):173–183, 2011.

[27] Tim Mattson (IL) and Rob van der Wijngaart (SSG).
RCCE: a Small Library for Many-Core Communication.
Intel Corporation, 2010.

[28] Wonyoung Kim, M.S. Gupta, Gu-Yeon Wei, and D. Brooks.
System level analysis of fast, per-core dvfs using on-chip switching regulators.
In *High Performance Computer Architecture, 2008. HPCA 2008. IEEE 14th International Symposium on*, pages 123–134, Feb 2008.

[29] Jean-Claude Laprie.
Dependable computing and fault-tolerance.
Digest of Papers FTCS-15, pages 2–11, 1985.

References V

[30] Yudan Liu, Raja Nassar, Chokchai Leangsuksun, Nichamon Naksinehaboon, Mihaela Paun, and Stephen L Scott.
An optimal checkpoint/restart model for a large scale high performance computing system.
In *Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on*, pages 1–9. IEEE, 2008.

[31] Alexandros Mavrogiannis.
On the dependability of transient neuron simulations.
Bachelor thesis, National Technical University of Athens, June 2014.

[32] Shubu Mukherjee.
Architecture Design for Soft Errors.
Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2008.

[33] S.S. Mukherjee, J. Emer, and S.K. Reinhardt.
The soft error problem: an architectural perspective.
In *High-Performance Computer Architecture, 2005. HPCA-11. 11th International Symposium on*, pages 243–247, Feb 2005.

[34] Venkatesh Pallipadi.
Enhanced Intel SpeedStep Technology and Demand-Based Switching on Linux*.
Technical report, Intel, 10 2010.

[35] Arun Raghavan, Laurel Emurian, Lei Shao, Marios Papaefthymiou, Kevin P. Pipe, Thomas F. Wenisch, and Milo M. K. Martin.
Computational sprinting on a hardware/software testbed.

References VI

[36] Arun Raghavan, Yixin Luo, Anuj Chandawalla, Marios Papaefthymiou, Kevin P. Pipe, Thomas F. Wenisch, and Milo M. K. Martin.
Computational sprinting.
In *Proceedings of the 2012 IEEE 18th International Symposium on High-Performance Computer Architecture*, HPCA '12, pages 1–12, Washington, DC, USA, 2012. IEEE Computer Society.

[37] D. Rodopoulos, F. Catthoor, and D. Soudris.
Tackling performance variability due to ras mechanisms with pid-controlled dvfs.
Computer Architecture Letters, PP(99):1–1, 2014.

[38] D. Rodopoulos, G. Chatzikonstantis, A. Pantelopoulos, D. Soudris, C.I. De Zeeuw, and C. Strydis.
Optimal mapping of inferior olive neuron simulations on the single-chip cloud computer.
In *Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS XIV), 2014 International Conference on*, pages 367–374, July 2014.

[39] D. Rodopoulos, A. Papanikolaou, F. Catthoor, and D. Soudris.
Demonstrating hw-sw transient error mitigation on the single-chip cloud computer data plane.
Very Large Scale Integration (VLSI) Systems, IEEE Transactions on, 23(3):507–519, March 2015.

[40] Luís Moura Silva and João Gabriel Silva.
System-level versus user-defined checkpointing.
In *Reliable Distributed Systems, 1998. Proceedings. Seventeenth IEEE Symposium on*, pages 68–74. IEEE, 1998.

[41] John Paul Walters and Vipin Chaudhary.
Application-level checkpointing techniques for parallel programs.
In *Distributed Computing and Internet Technology*, pages 221–234. Springer, 2006.

References VII

[42] Xuejun Yang, Zhiyuan Wang, Jingling Xue, and Yun Zhou.
The reliability wall for exascale supercomputing.
Computers, IEEE Transactions on, 61(6):767–779, June 2012.