Optimal mapping of inferior olive neuron simulations on the Single-Chip Cloud Computer

Conference Paper · July 2014

DOI: 10.1109/SAMOS.2014.6893235

6 authors, including:

Dimitrios Soudris
National Technical University of Athens
598 PUBLICATIONS **2,831** CITATIONS

SEE PROFILE

Chris I De Zeeuw
Erasmus University Rotterdam
478 PUBLICATIONS **20,321** CITATIONS

SEE PROFILE

Optimal Mapping of Inferior Olive Neuron Simulations on the Single-Chip Cloud Computer

Dimitrios Rodopoulos, Giorgos Chatzikonstantis,
Andreas Pantelopoulos and Dimitrios Soudris
MICROprocessors and digital systems LABoratory, NTUA, Greece
{drodo, georgec, pantelopoulos, dsoudris}@microlab.ntua.gr

Chris I. De Zeeuw and Christos Strydis
Department of Neuroscience, Erasmus MC
Rotterdam, The Netherlands
{c.dezeeuw, c.strydis}@erasmusmc.nl

***Abstract*—Biologically accurate neuron simulations are increasingly important in research related to brain activity. They are computationally intensive and feature data and task parallelism. In this paper, we present a case study for the mapping of a biologically accurate inferior-olive (InfOli), neural cell simulator on an many-core research platform. The Single-Chip Cloud Computer (SCC) is an experimental processor created by Intel Labs. The target neurons provide a major input to the cerebellum and are involved in motor skills and space perception. We exploit task- and data-partitioning, scaling the simulation over more than 40,000 neurons. The voltage- and frequency-scaling capabilities of the chip are explored, achieving more than 20% energy savings with negligible performance degradation. Four platform configurations are evaluated and a mapping with balanced workload and constant voltage and frequency is formally derived as optimal.**

***Keywords*—Dynamic Frequency and Voltage Scaling, Inferior Olive Neurons, Pareto Optimal, Single-Chip Cloud Computer**

I. Introduction

In-vitro and in-vivo neuroscientific experiments are costly, time-consuming and may require testing on animals. They are also highly complex, often difficult to reproduce and offer limited tissue access. Hence, constructing and exploring realistic simulations of neural networks on computing platforms is a viable and useful alternative for neuroscientists [1], [2]. The resulting discipline of neuroinformatics comprises a powerful tool in the hands of the community. However, in order to accurately simulate brain-cell activity, one has to properly cope with various biological aspects, prominently a *large neural-cell population* and also a complex *connectivity scheme between the simulated neurons*[1]. Both attributes call for significant computational resources with high inherent parallelism. Thus, they are very much in-line with the multi-/many-core paradigm observed in modern computer infrastructure trends [3], [4].

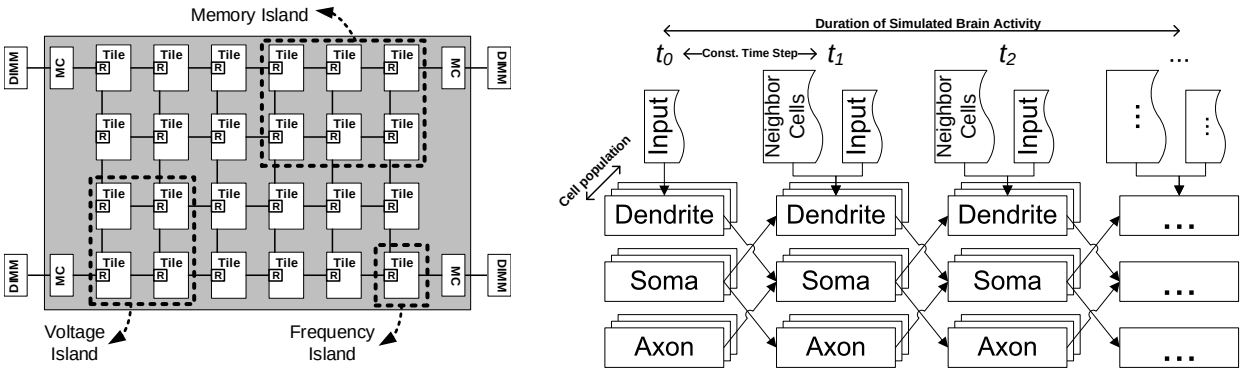
A wide variety of neuron models has been developed, targeting different levels of detail [5], [6], [7], popular among which are the time driven (extended) Hodgkin-Huxley models. They are solved in a transient way and provide the voltage response of neurons, given user-defined current stimuli. Platforms with many execution units are typically being selected to solve these models, given their inherent data parallelism [8], [9], [10]. Despite the plethora of models and execution platforms, there has been reduced emphasis on systematically optimizing the model execution on a many-core platform. Given the capabilities of modern computing systems, the degrees of freedom that are available to the community allow an aggressive exploration of the performance vs. quality-cost trade-off. This enables cost-conscious execution of biologically accurate neural models on multi-/many-core systems. There is significant potential in using such systematic approaches, given the magnitude and energy budget [11] of computing infrastructure currently employed for brain modeling [12].

In the current paper, we study the performance vs. quality-cost trade-off of the InfOli simulator [13], [14] on a many-core chip. The SCC experimental processor [15] is a 48-core "concept vehicle" created by Intel Labs as a platform for many-core software research. It features Dynamic Voltage and Frequency Scaling (DVFS) and on-die message passing. A systematic Design Space Exploration (DSE) produces the optimal application mapping. We prove that a symmetric mapping (i.e. identical workload per core) with static voltage/frequency configuration is optimal in terms of performance and energy. The current paper is organized as follows: Section II discusses prior art. In Section III we present attributes of the target platform and application. In Section IV we partition the InfOli simulator and propose power management strategies. The derived design space is explored in Section V to derive optimal mapping. Finally, conclusions are drawn in Section VI.

II. Related Work & Motivation

With respect to simulating biological phenomena, neural-cell models encompass different degrees of accuracy [16]. An exhaustive presentation of neuromodeling approaches is beyond the scope of this paper, however, we briefly introduce some representative prior art. According to O' Reilly et al. [17], there are different "hierarchical levels of analysis" to confront the brain's extreme complexity. They mention that one could ignore the "underlying biological mechanisms of cognition", focusing instead on patterns and algorithmic computations the brain performs when undertaking a task. They focus on neural networks, using methods from the field of artificial intelligence to explore the process of human thinking. Other approaches are really abstract, as in the case of traditional neural networks [18]. In this case, a set of neurons is abstracted by processing elements with different weights, the values of which are combined to give a single output. This approach is useful for solving non-linear control problems. In a process that is called *training*, the weights of each processing element are selected and the desired output for typical use cases is derived. However, this approach exposes almost no biological information of the considered "neurons".

[1]The terms *neuron* and *(neural) cell* will be used interchangeably.



(a) The frequency, voltage and memory islands of the SCC

(b) Illustration of the InfOli simulator runtime

Fig. 1: High level view of the target platform and the target application that are discussed in this paper

In comparison to state-of-the-art approaches, the model that is used in our work [13], [14] is of *higher accuracy from a biological perspective*. It is an extended version of the Hodgkin-Huxley [5] model, which captures the potential and current of (neural) cell compartments, based on a juxtaposition of biological parameters with the electrical characteristics of an equivalent circuit. The resulting model falls under the Spiking-Neural Network (SNN) class of models [19], [20] allowing transient inspection of neural activity. Biologically accurate simulation of neural activity requires considerable computational resources. Execution on a set of processing elements is a typical course of action. Prior art features examples using Field-Programmable Gate Arrays (FPGAs) [8], multi-processors [10] or Graphics Processing Units (GPUs) [9]. Many of these approaches target real-time simulation, at the cost of limited cell populations. In the current paper, we deviate from this practice and aim to *scale the simulation up to a very large neuron population in an energy-conscious way*. A very important differentiator of the current paper from prior art is the *formal treatment* of the available design-time options. Each neuron simulation presented in this paper is assessed in terms of performance and energy budget. The majority of related works focuses entirely on the computational efficiency of neural simulations [21], rather than on the capabilities of the target platform. In our work, we maintain a system-level viewpoint and derive different combinations of mapping and power-management, creating a design space of platform configurations. This space is formally treated to identify the optimal platform configurations for the InfOli simulator.

III. Target Platform and Application

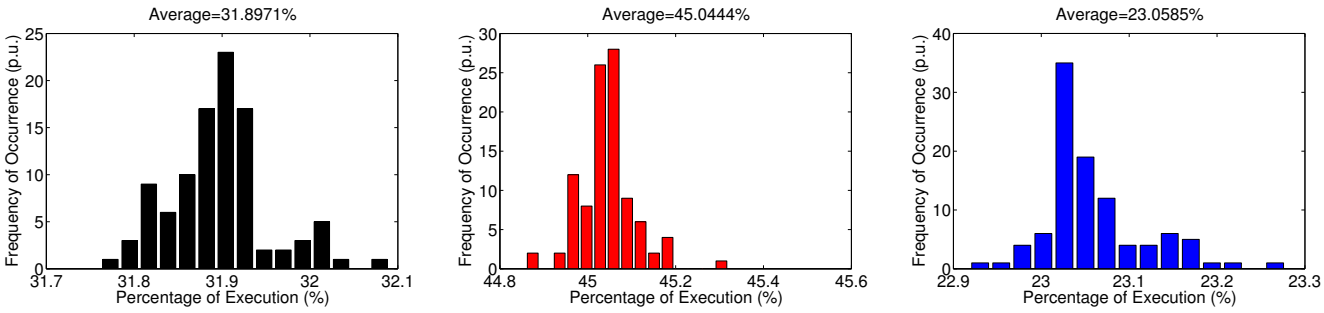
This Section discusses the mapping of the InfOli simulator on the SCC chip. We briefly present the target platform (Subsection III-A). The run-time of the simulator is discussed and a profiling of the execution time is provided (Subsection III-B). Subsection III-C discusses neuron inter-connectivity and formulates a probabilistic model for neuron communication.

A. Overview of the SCC Platform

The Intel SCC [15] is a homogeneous, many-core chip with 48 cores, organized in pairs called *tiles*, which are interconnected through a mesh network (Figure 1a). Each tile constitutes a separate *frequency island*. Four tiles create an individual *voltage island*. The default configuration assigns each quarter of the chip to a Dual Inline Memory Module (DIMM) through a Memory Controller (MC). Each core of the SCC uses a low-level Message-Passing Buffer (MPB) used for the exchange messages with other cores. In the context of the current paper, each core is booted with a custom Linux distribution. The SCC chip is installed on a board that communicates with a Management-Console Personal Computer (MCPC) through Ethernet (for power monitoring) and PCIe (for disk access) connections. The `/shared` directory is common between the MCPC and the SCC cores, thus enabling exchange of files and executables. Source code is written and cross-compiled on the MCPC and executed on the SCC. We use the default programming library of the SCC, called RCCE, for message passing and DVFS [22]. Tasks are dispatched from the MCPC to the SCC and the total chip power can be recorded from the SCC Board Management Controller (BMC) over telnet [23].

B. Overview of the InfOli Simulator Runtime

For a given cluster of InfOli neurons, the simulator calculates the membrane potential of each cell under the influence of external and internal input stimuli (currents). The simulator is *transient* (i.e. time driven) with a constant step equal to 50 μs. The application flow during simulation steps $t_0, t_1, t_2, ...$ can be seen in Figure 1b. A very important element is the *inter-connectivity* between the cells. In order to realistically calculate the potential of each cell, the simulator needs to consider the voltage levels of neighboring cells, to which the former cell is connected. As a result, the degrees of freedom of the simulation, which should also be specified by the user before the simulation starts, are: (i) the *size of the cell network*, (ii) the *duration of the simulated brain activity*, (iii) the *external input currents to each cell* as well as the desired (iv) *interconnectivity scheme*. The simulator solves the model for each of the simulated neurons, assuming three compartments per cell [24]: (i) The *dendrite* compartment exchanges information with other InfOli cells and receives the input current at the beginning of each simulation step. (ii) After input current stimuli have been received from the environment, the *soma* compartment performs the most computationally intensive part of the neuron. (iii) The *action potential* of the *axon* component constitutes the "output" of each neuron cell.



(a) Profiling of the dendrite compartment (b) Profiling of the soma compartment (c) Profiling of the axon compartment

Fig. 2: Profiling on a single SCC core (at default frequency/voltage setting), to extract timing information for each compartment

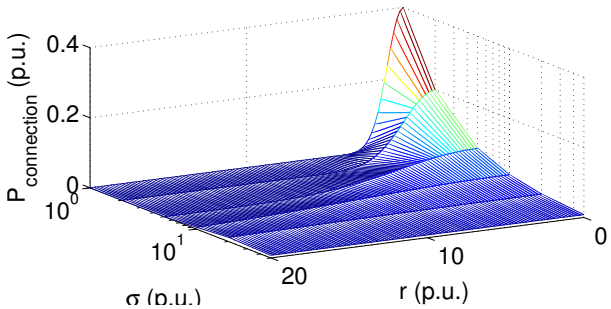


Fig. 3: Connection probability according to Equation 1

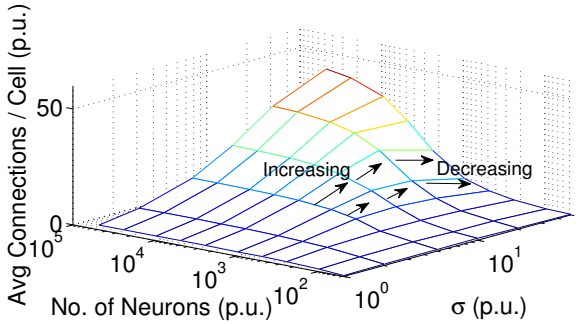


Fig. 4: Drop in mean number of connections after a certain σ.

The simulator is a system *with memory*, since the state of each compartment is reused in the next simulation iteration. We start from a single-threaded implementation that includes these three cell compartments for a single InfOli neuron cell and perform a profiling experiment to derive the workload distribution between the three compartments. The results are shown in Figure 2 for 100 runs of the same simulation for 6 seconds of brain activity simulated on a single SCC core with constant voltage and frequency. The UNIX function `getttimeofday` has been used to record execution times. Immediately, we verify that the soma compartment is the most computationally intensive part of the simulator's runtime. The axon compartment exhibits the shortest processing times. The dendrite component lies in the middle. However, given that it is related to inter-neuron communication, we expect that it will occupy a large fraction of the overall execution time in case we employ a more complicated interconnectivity scheme.

$$P_{\text{connection}} = f(r, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{r^2}{2\sigma^2}}, \text{ where } r > 0 \quad (1)$$

C. Replicating Neuron Inter-Connectivity

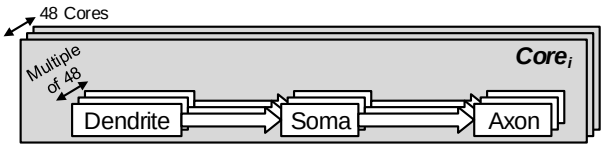
In order to realistically simulate neuron behavior, it is important to replicate not only the computation but also the inter-connection between neurons. Initially, we assume that the simulated neurons are organized in a two dimensional mesh. Then, we create inter-neuron connections in a probabilistic way, based on the probability density function of the normal distribution. Thus, assuming two neurons, at a distance r, the probability of their being connected is given by Equation 1, where σ describes the dispersion of inter-neuron connections across the mesh (Figure 3). In the context of the current paper, we assume a two dimensional mesh of neurons, hence $r = \sqrt{x^2 + y^2}$, where x and y are relative cartesian coordinates. Both r and σ are quantified in numbers of neurons (p.u).

It should be noted that, for a specific network size, an increasing σ does not necessarily increase the average number of connections per neuron. This is highlighted in Figure 4: Up to a specific value of σ, the average number of connections is indeed increasing. However, after that point, the average number of connections is decreasing, since the connection spread is very wide for the target finite neuron inventory.

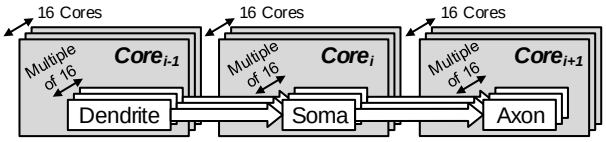
IV. Simulator Mapping & Power Management

A. Partitioning the InfOli Simulator

1) Data Partitioning: If we imagine a plane of InfOli neurons, the most intuitive mapping option of the respective model would be to assign a subset of the cell population to each of the cores of a many-core system. This effectively represents *data partitioning* of the simulation and it will be the first mapping strategy explored in this paper. We shall only instantiate network sizes that are multiple of 48 to the SCC chip, in order to maintain a homogeneous distribution of workloads across the SCC cores. This mapping is illustrated in Figure 5a. With respect to inter-cell communication, cells assigned to the same core communicate through its private memory. Cells assigned to different cores communicate over the MPB with the RCCE message passing commands [22].



(a) Data partitioning, where $i = 0, 1, 2, ..., 47$



(b) Combined data and task partitioning, where $i = 1, 4, ..., 46$

Fig. 5: The two mapping options of the InfOli neuron cell simulator on the SCC chip that are discussed in this paper

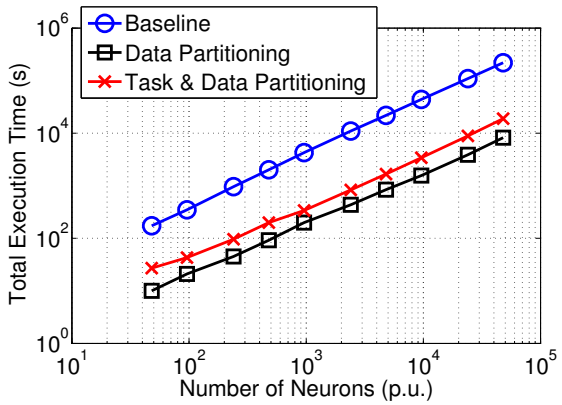
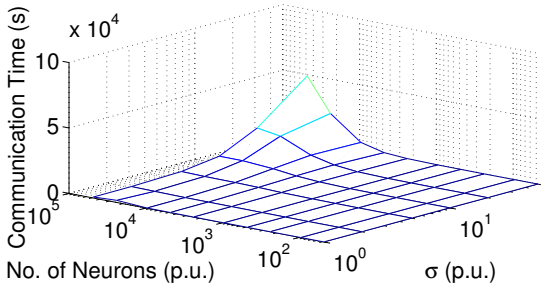


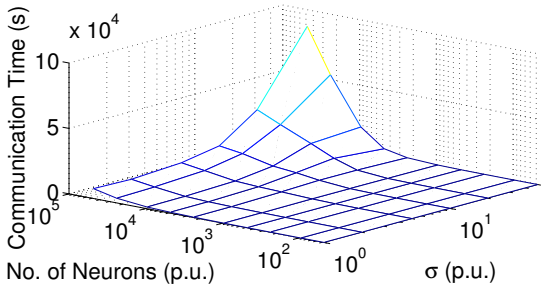
Fig. 6: Initial benchmarking (V_{dd} =1.1 V, f = 533 MHz, $\sigma = 1$); baseline represents execution on a single SCC core.

2) Task & Data Partitioning: From Figure 2, we can approximate the relative execution times of the three neuron compartments. The soma requires on average twice the execution time, in comparison to the dendrite and axon. Thus, it is reasonable to initially split the simulation with respect to neuron compartments. Assigning the simulation of each compartment to a different core effectively constitutes *task partitioning*. With a total of 48 available cores, we additionally perform data partitioning, thus implementing *combined task and data partitioning* on the InfOli neuron simulator, illustrated in Figure 5b. We assign each compartment to one core, thus an InfOli neuron requires a triplet of cores. Each core simulates one compartment of $N \div 16$ cells, where N is the size of cell network (multiple of 16, to avoid workload imbalance).

3) Initial Findings: In Figure 6, we see the execution times for the two considered partitioning schemes. Both have been verified against the Baseline (execution on a single SCC core) and show no accuracy degradation. We achieve a constant speedup of about an order of magnitude. Combined data and task partitioning exhibits a lower speedup. This is to be expected, due to the increased inter-core traffic required for *both* inter-cell *and* inter-compartment communication applying to this partitioning: the execution time dedicated to message passing time is larger in the combined task and data partitioning case (Figure 7b), compared to data partitioning (Figure 7a).



(a) Data Partitioning



(b) Task & Data Partitioning

Fig. 7: Impact of non-regular neuron inter-connectivity (based on Equation 1) on the message passing overhead. The SCC chip is constantly trained at $V_{dd} = 1$ V and $f = 533$ MHz.

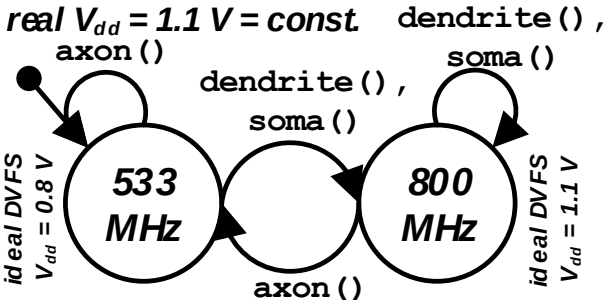


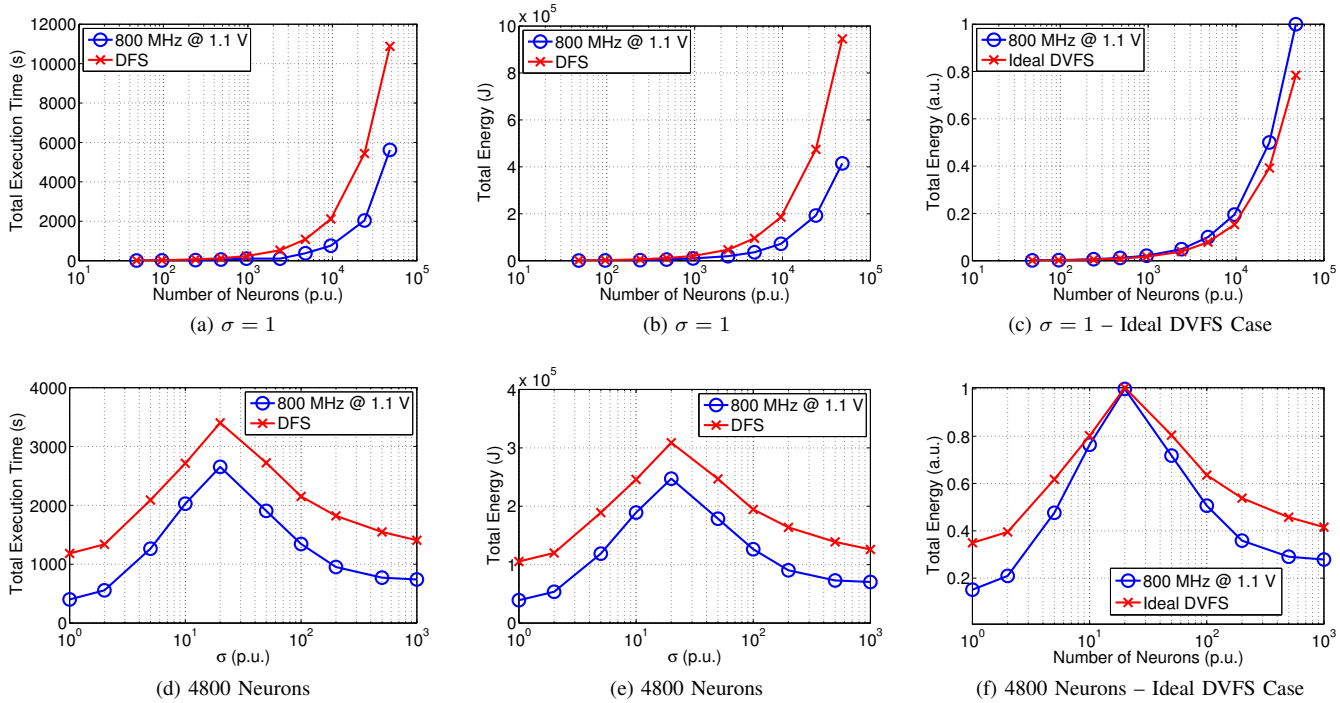
Fig. 8: The DFS FSM for the data partitioned InfOli simulator. Voltage values for the ideal DVFS scheme are also highlighted.

B. Power Management of the InfOli Simulator

As a transient InfOli simulation contains more cells, the duration of a simulation may exceed tens of hours. Thus, it attractive to explore power or energy minimization techniques. The SCC enables the adjustment of voltage and frequency at a granularity illustrated in Figure 1a[2]. It also allows transient monitoring of the total power consumed by the SCC chip. For the rest of this paper, and without loss of generality, we deactivate all simulator interim output messages. Thus, the simulator reports only the final status of the simulated neurons.

1) Data Partitioning: Based on the partitioning option of Subsubsection IV-A1, a power-management strategy should be applied that is global across the SCC chip. As the profiling

[2]We note that prior work has stated that a constant maximum frequency is not necessarily the most energy efficient [25]. However, confirmation or rebuttal of this statement falls beyond the scope of the current paper.



(a) $\sigma = 1$ (b) $\sigma = 1$ (c) $\sigma = 1$ – Ideal DVFS Case

(d) 4800 Neurons (e) 4800 Neurons (f) 4800 Neurons – Ideal DVFS Case

Fig. 9: Performance and quality cost analysis for the data partitioning of the InfOli simulator

information has revealed (Figure 2), the soma compartment requires roughly twice the execution time taken up by the axon or dendrite compartments. The dendrite component is a bit faster in comparison to the soma, however given a more complicated inter-connectivity scheme, its computational overhead is bound to increase. If we were to be conservative with the clock frequency of each core, an interesting opportunity would arise: Given the Dynamic-Frequency-Scaling (DFS) capabilities of the SCC (at tile granularity), we can manipulate the frequency of the cores depending on the compartment that they are simulating in each case. We chose to alternate between the 533 MHz and 800 MHz clock frequencies, by properly setting the frequency divider of each tile. The Finite-State Machine (FSM) of Figure 8 is illustrating this concept. We avoid voltage manipulation at runtime, since the SCC voltage regulators are less responsive than the frequency dividers [26]. Since both frequencies need to be supported at runtime, the voltage supply is set at the minimum of 1.1 V. We also test a constantly high clock (800 MHz) with the lowest allowed voltage (1.1 V).

We measure the execution time from the MCPC and total energy by signaling the BMC of the chip[3]. Each measurement session covers 6 second simulation of brain activity, various cell populations (Figure 9a and 9b) and certain inter-connectivity dispersion (Figure 9d and 9e). A constant, high frequency is a clearly the preferable choice. As expected, the DFS option is worse both in terms of performance and energy. In case the voltage regulator could be of the same responsiveness as the frequency divider, we could witness actual energy benefits. Assuming such *ideal DVFS*, following the FSM of Figure 8, we can estimate the total energy, using the execution times of Figures 9a and 9d, as well as the square law of power consumption $P = cfV_{dd}^2$, where c is a parameter for the switching capacitance of the chip [28]. Given that this is unknown for the SCC, we present the energy estimation in arbitrary units, normalized for the maximum energy value. The result are shown in Figures 9c and 9f. In the case of the neuron count sweep (Figure 9c), we can clearly see how DVFS trades off a portion of performance for a more energy efficient execution (by a maximum of 20%). In the case of the inter-connectivity sweep (Figure 9f). However, given that frequency manipulation imposes a constant overhead (Figure 9d) and the total execution time is upper-bounded by a maximum of inter-connectivity (Subsection III-C), the ideal case of DVFS yields no energy benefits whatsoever. With a higher degree of inter-connectivity temporal overhead (e.g. larger neuron network) we may have had energy benefits in the case of ideal DVFS.

2) Task & Data Partitioning: In the case of combined task and data partitioning (discussed in Subsubsection IV-A2), we can have different power-management strategies per core since the cores are executing different tasks. We isolate tasks that simulate specific compartments and map them to separate voltage islands of the chip [29]. Based on the profiling information collected previously, voltage islands simulating somata or dendrites need to operate at twice the frequency of voltage islands simulating axons. Dendrites are assigned to high-frequency execution due to the potential for intensive inter-core communication, in case complex inter-connectivity schemes are simulated. Hence, cores that need to run "fast" will be set to 800 MHz and "slow" cores will run at 533 MHz. Frequency scaling needs to be performed only once before the

[3]Prior art indicates instabilities in the power measurement of the SCC [27]. Considering that such fluctuations are in the vicinity of a 1 W and given the averaging-out of numerical integration for energy calculation, finer granularity of SCC power measurement is beyond the scope of the current paper.

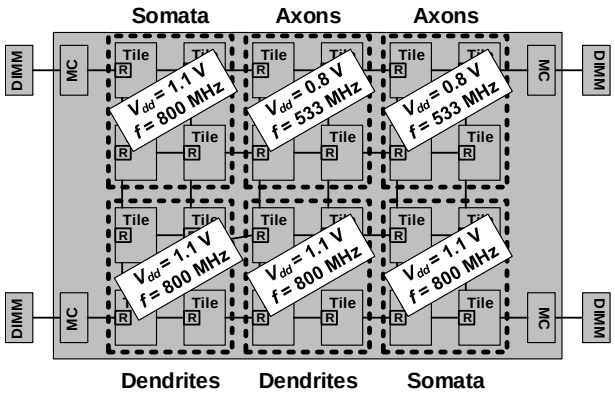
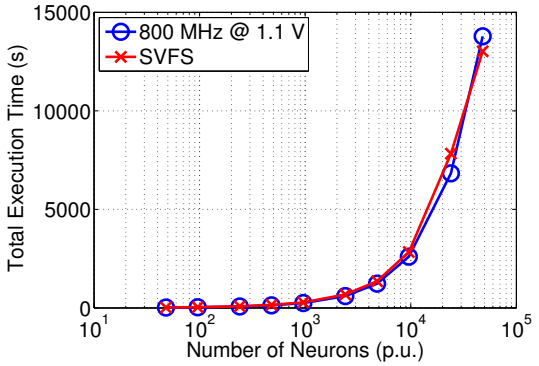


Fig. 10: SVFS for the combined task and data partitioning

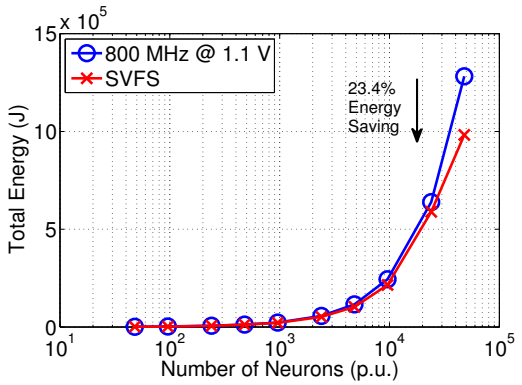
simulation begins. That way, we have the chance to (statically) "train" the SCC system to the appropriate voltages. "Fast" voltage islands will be configured at 1.1 V (minimum voltage setting to support a 800 MHz clock) and "slow" voltage islands will run at 0.8 V (minimum voltage that supports a 533 MHz clock). We refer to this power management as Static Voltage and Frequency Scaling (SVFS), shown in Figure 10. Given the different frequencies and voltages used by the SVFS setting, it is important to average out the stressing of the cores of the SCC (i.e. avoid hot spots or heterogeneous aging across the many-core chip). Addressing this issue is beyond the scope of the current paper, however solutions have been already proposed in the literature [30]. We also test a globally high clock scheme, where all islands are trained to 800 MHz and 1.1 V. In a second set of measurements, we present the execution time and total energy for various neuron populations (Figure 11a and 11b) and degrees of inter-connectivity (Figure 11c and 11d), assuming the same 6 second brain activity. Apparently, SVFS is a very successful power management scheme for combined task and data partitioning. It saves significant energy (more than 20%) at a negligible performance degradation. Especially for 48,000 neurons with $\sigma = 1$, the performance is almost similar to the case of uniform power management. Finally, in Figures 9d–9e for data partitioning and Figures 11c–11d for combined task and data partitioning, we see that the performance and quality cost of the simulator drops after a certain σ value. This is to be expected, if we consider the probabilistic connectivity model of Equation 1. In Figure 4 we have highlighted that when the neuron inter-connection spread is very wide for the target cell population, the amount of formed connections drops. Thus, the inter-neuron communication (and hence cores of the SCC) reduces, with a consequent drop in execution time and energy. In any case, SVFS provides energy benefits around 15% in the case of combined task and data partitioning. At the same time negligible performance degradation is created to the simulator.

V. Formal Treatment of the Design Space

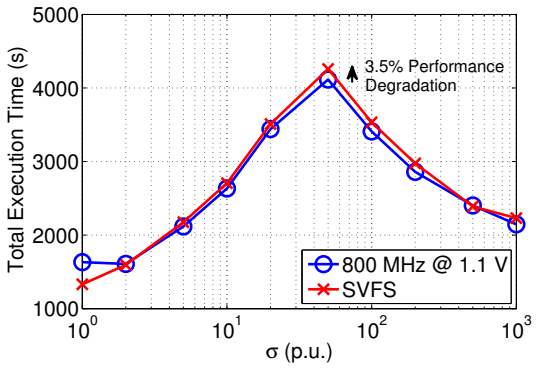
The variety of design-time choices that can be made when mapping the InfOli simulator on the SCC create a set of *system configurations*. Each one comes as a different blend of task/data partitioning and represents a specific power-management strategy. Based on Section IV, the available configurations are summarized as: {800 MHz @ 1.1 V – Task



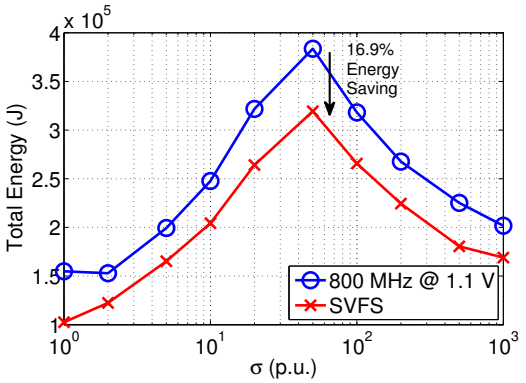
(a) Task & Data Partitioning: $\sigma = 1$



(b) Task & Data Partitioning: $\sigma = 1$



(c) Task & Data Partitioning: No. of Simulated Neurons = 4800



(d) Task & Data Partitioning: No. of Simulated Neurons = 4800

Fig. 11: Performance and quality cost analysis of the combined task and data partitioning of the InfOli simulator

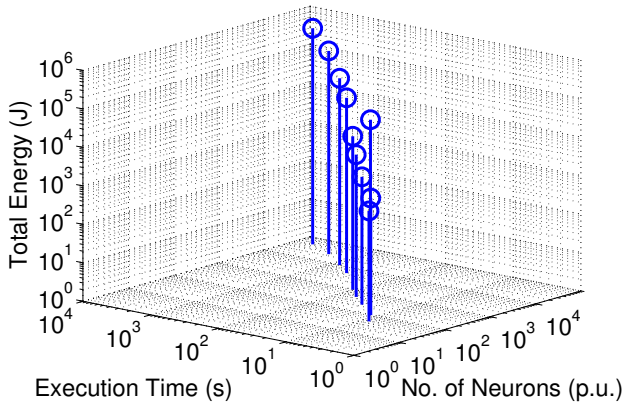
& Data Partitioning, SVFS – Task & Data Partitioning, 800 MHz @ 1.1 V – Data Partitioning, DFS – Data Partitioning}. Each configuration can be evaluated in terms of certain *cost metrics*, such as execution time or consumed energy, producing a single *design point*. This set of points creates a *design space* which can be treated formally to derive *Pareto-optimal* [31] design points. It is noted that the hypothetical exploration of Figures 9c and 9f is not included in the DSE presented here.

The target neuron network size, the inter-connectivity scheme and the duration of simulated brain activity are the independent variables of this exploration. Initially, values for these three variables are defined. Then, a set of n available design points is identified. Each design point P_i is evaluated against cost metrics $x_j(P_i)$, where $j = 1, 2, ..., m$. The front of Pareto optimality is a set V with all the design points P_i that satisfy Equation 2, for n design points and m cost metrics:

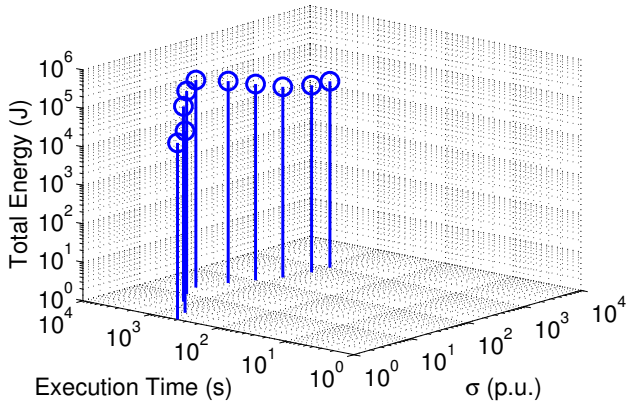
$$x_j(P_i) \leq x_j(P_k), \quad \forall\ k = 1, 2, ..., n \text{ and } \forall\ j = 1, 2, ..., m \tag{2}$$

The derivation of the Pareto front can be automated using algorithms that calculate minima of a set of vectors [32]. To accelerate exploration, the Pareto front can be pruned further if constraints are imposed on cost metrics. For example, in *real-time* InfOli-neuron simulation, the total execution time should not exceed the duration of simulated brain activity. To perform the aforementioned methodology, knowledge of the InfOli-simulator design points is required. This can originate from a extensive benchmarking session at design time. In order to substantiate the proposed DSE methodology, we identify the Pareto front of optimal InfOli-simulator mappings for two different cases. The duration of simulated brain activity is assumed constant in both cases, equal to 6 seconds. In the first case we assume a constant $\sigma = 1$ and go through different sizes of neuron network. In the second case, we fix the number of neurons to 4,800 cells and make a sweep of possible σ values. The results are shown Figures 12a and 12b. In both cases, data partitioning at 800 MHz / 1.1 V is solely derived as optimal.

From the above findings, we see that a symmetric mapping is preferable than the introduction of inter-core load imbalance through task partitioning. Even if the latter leads to V_{dd} reduction for certain voltage islands (SVFS–Subsubsection IV-B2), it is much preferable to maintain a symmetric approach, where all cores executed the same workload at constant voltage and frequency. This is compatible with the Single Program - Multiple Data (SPMD) programming style of the SCC [22]. The explored DFS scheme cannot, by nature, provide energy benefits. However, it can reveal an upper bound to the energy benefits of a highly responsive (though, not materialized in the case of the SCC) DVFS utility (see Figures 9c and 9f). Finally, we can strongly motivate that in order to provide performance vs. quality cost optimality guarantees, we need to perform such a DSE (a general view is provided in Figure 13). The user (i.e. a neuroscientist) specifies the neuron network size, interconnectivity and duration of the simulation. Based on the DSE, the developer (i.e. the platform programmer) adapts the user model to a Pareto-optimal configuration for the target many-core system (i.e. mapping and power-management options). Thus, *any experiments of the user are bound to occupy the target platform in a Pareto-optimal way*.



(a) Pareto points for $\sigma = 1$ and various neuron network sizes



(b) Pareto points for 4,800 neurons with various inter-connectivity densities

Fig. 12: Two DSE results for the InfOli simulator (constant σ and constant number of neurons) for 6 seconds of brain activity

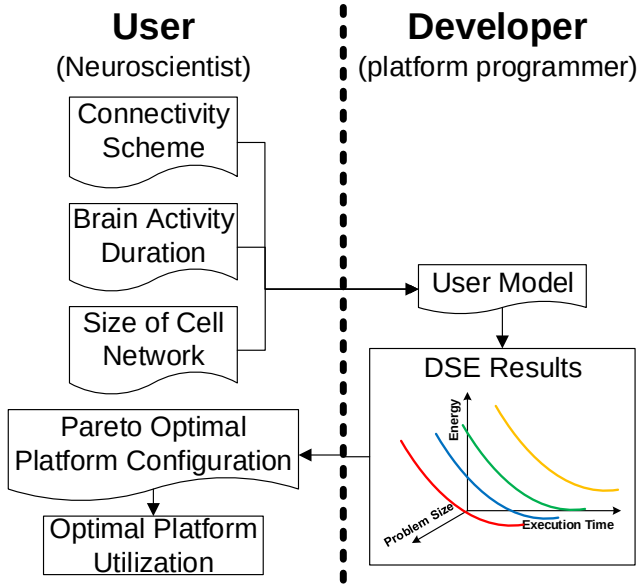


Fig. 13: DSE methodology for the optimal mapping of the InfOli neuron simulator on a many-core platform

VI. Conclusion

In this paper, we have presented a thorough DSE for the mapping of a biologically accurate neuron simulator on an industrial grade many-core platform. The simulator of our choice is based on a transient, time driven model for the inferior-olive neurons, which are of major importance for human sensorimotor control. The target platform is the Single-Chip Cloud Computer, developed by Intel Labs. In the feasibility study presented herein, we have explored different partitioning schemes, based on data and combined task-and-data partitioning. Also, we explored the power-management options of the chip, implementing both Dynamic Frequency Scaling and Static Voltage and Frequency Scaling. Combinations of mapping and power-management options create a design space of different points. The quality cost of the simulation along with the sensitivity of the Pareto space in problem parameters, motivate a systematic treatment of this design space, in order to guarantee truly optimal utilization of the platform. A Pareto optimality problem has been formulated to extract such optimal platform configurations. The findings of the DSE reveal that a symmetric configuration, with identical workload-per-core and a global power management policy is optimal for the mapping of the InfOli simulator on the SCC.

Acknowledgment

The SCC is available under an MTA between ICCS/NTUA and Intel Corporation. This work is partially supported by the HiPEAC 3 EU ICT-287759 Collaboration Grant and by the FP7-287611-DeSyRe and FP7-612069-HARPA EU projects.

References

[1] Davison A.P. et al., "Trends in programming languages for neuroscience simulations," *Frontiers in Neuroscience*, vol. 3:3, pp. 374–380, 2009.

[2] Zaytsev Y.V. et al., "Increasing quality and managing complexity in neuroinformatics software development with continuous integration," *Frontiers in Neuroinformatics*, vol. 3:3, pp. 6–31, 2013.

[3] Fuller, S.H. et al., "Computing Performance: Game Over or Next Level?" *Computer*, vol. 44, no. 1, pp. 31–38, 2011.

[4] H. Markram, "The Blue Brain Project," *Nature Reviews Neuroscience*, vol. 7, no. 2, pp. 153–160, 2006.

[5] Hodgkin, A. L. et al., "A quantitative description of membrane current and its application to conduction and excitation in nerve," *Journal of Physiology*, vol. 117, no. 4, 1952.

[6] E. Izhikevich, "Simple model of spiking neurons," *Neural Networks, IEEE Transactions on*, vol. 14, no. 6, 2003.

[7] I. Segev, Ed., *Methods in Neuronal Modeling - 2nd Edition: From Ions to Networks*. A Bradford Book, 1998.

[8] Thomas, D. B. et al., "FPGA Accelerated Simulation of Biologically Plausible Spiking Neural Networks," in *FCCM*, 2009, pp. 45–52.

[9] Yamazaki, T., "GPU-based implementation of a cerebellar spiking network model for realtime robot control," in *21st Annual Conference of the Japanese N.N. Society*, 2011.

[10] Jin, X., "Efficient modelling of spiking neural networks on a scalable chip multiprocessor," in *IJCNN*, 2008.

[11] Subramaniam, B. et al., "The Green Index: A Metric for Evaluating System-Wide Energy Efficiency in HPC Systems," in *IPDPSW*, 2012.

[12] École Polytechnique Fédérale de Lausanne. (2012, June) The Blue Brain Project – Main Components of the Infrastructure.

[13] De Gruijl, J. R. et al., "Climbing Fiber Burst Size and Olivary Subthreshold Oscillations in a Network Setting," *PLoS Comput Biol*, 2012.

[14] Bazzigaluppi, P. et al., "Olivary subthreshold oscillations and burst activity revisited," *Frontiers in Neural Circuits*, vol. 6, no. 91, 2012.

[15] Howard, J. et al., "A 48-Core IA-32 Processor in 45 nm CMOS Using On-Die Message-Passing and DVFS for Performance and Power Scaling," *IEEE JSSC*, vol. 46, no. 1, pp. 173–183, 2011.

[16] Dayan, P., *Levels of Analysis in Neural Modeling*. Wiley, 2006.

[17] O'Reilly, R. et al., *Computational Explorations in Cognitive Neuroscience: Understanding the Mind by Simulating the Brain*. Bradford.

[18] Nguyen, D.H. et al., "Neural networks for self-learning control systems," *Control Systems Magazine, IEEE*, vol. 10, no. 3, 1990.

[19] Bishop, C. M. et al., Ed., *Pulsed Neural Networks*. Bradford, 2001.

[20] Gerstner, W. et al., *Spiking Neuron Models*. Cambridge University Press, 2002.

[21] Agudelo-Toro, A. et al., "Computationally efficient simulation of electrical activity at cell membranes interacting with self-generated and externally imposed electric fields," *Jour. of Neural Engineering*, 2013.

[22] Mattson, T.G. et al., "The 48-core SCC Processor: the Programmer's View," in *Int. SC Conf.*, nov. 2010, pp. 1 –11.

[23] Intel Labs, "The SCC Platform Overview," Tech. Rep. Rev 0.7, 2010.

[24] De Zeeuw, C. I. et al., "Spatiotemporal firing patterns in the cerebellum," *Nature Review Neuroscience*, vol. 12, no. 6, pp. 327–344, 2011.

[25] P. Gschwandtner, T. Fahringer, and R. Prodan, "Performance analysis and benchmarking of the intel scc," in *Cluster Computing (CLUSTER), 2011 IEEE International Conference on*, Sept 2011, pp. 139–149.

[26] Intel Labs, "The SCC Programmer's Guide–0.75," Tech. Rep., 2010.

[27] Bakker, R. et al., "Emulating asymmetric mpsocs on the intel scc many-core processor," in *PDP–Euromicro*, Feb 2014, pp. 520–527.

[28] D. Soudris, C. Piguet, and C. Goutis, Eds., *Designing CMOS Circuits for Low Power*. Springer, 2002.

[29] "Using the RCCE Power Management Calls – Revision 1.1," Intel Corporation, Tech. Rep., September 2011.

[30] Karpuzcu, Ulya R. et al., "The BubbleWrap many-core: popping cores for sequential acceleration," in *MICRO*, 2009, pp. 447–458.

[31] Geilen, M. et al., "An Algebra of Pareto Points," *Fundam. Inf.*, vol. 78, no. 1, pp. 35–74, Jan. 2007.

[32] Kung, H. et al., "On Finding the Maxima of a Set of Vectors," *J. ACM*, vol. 22, no. 4, pp. 469–476, Oct. 1975.