Proceedings of the 2020 Winter Simulation Conference
K.-H. Bae, B. Feng, S. Kim, S. Lazarova-Molnar, Z. Zheng, T. Roeder, and R. Thiesing, eds.

ENERGY EFFICIENCY EVALUATION OF PARALLEL EXECUTION OF DEVS MODELS IN MULTICORE ARCHITECTURES

Guillermo G. Trabes

Department of Systems and Computer Engineering
Carleton University
and Universidad Nacional de San Luis
1125 Colonel By
Ottawa, ON K1S 5B6, CANADA

Veronica Gil Costa

Universidad Nacional de San Luis
and CCT CONICET San Luis
Ejército de Los Andes 950
San Luis, D5700, ARGENTINA

Gabriel A. Wainer

Department of Systems and Computer Engineering
Carleton University
1125 Colonel By
Ottawa, ON K1S 5B6, CANADA

ABSTRACT

Complex models in science and engineering need better techniques to execute simulations efficiently. As we need high performance computers with many processors and memory to execute complex simulations faster, we face a problem with the consumption of energy. Therefore, we need new ways to define efficiency in simulations, measured not only in terms of computation time, but in terms of the amount of energy required to execute them. The Discrete-Event System Specification (DEVS) formalism, a well-known technique for modeling and simulation, includes simulation algorithms that allow running DEVS models in parallel computers. Nevertheless, no studies exist on the execution of DEVS simulations on parallel computers efficiently in terms of energy use. In this work, we show an energy efficiency evaluation of the execution of DEVS simulations on a shared-memory multicore architecture. The results presented show that executing in parallel can improve energy efficiency in these architectures.

1 INTRODUCTION

Discrete Event System Specification (DEVS) (Zeigler et al. 2000) is a well know formalism for modeling and simulation where models are described in a hierarchical and modular manner. DEVS has been used for modeling and simulations of systems in multiple fields of study and there has been a growing demand to simulate complex models, leading to increasing execution times. There have been multiple attempts to achieve parallel executions of DEVS using parallel discrete-event simulation approaches, but, in practice, the resulting simulation architectures end up being very complex and several issues arise related to zero lookahead loops and correctness.

In (Zeigler 2017), the author introduces a high-level DEVS simulation protocol for parallel execution, allowing multiple simultaneous transitions in the model to execute in parallel, while guaranteeing the correctness on the execution of the simulation. However, this protocol has not been tested with empirical results on parallel computers. Nowadays, parallel computers are composed by multithreaded systems.

Multithreaded systems have become popular due to the efficient use of systems resources and low communication latency compared to distributed memory systems. Nevertheless, many challenges arise in the programming of these architectures: race conditions, synchronization and coordination problems, and resource allocation and management. In addition, time performance and energy consumption are two important characteristics of applications, especially when they are executed in parallel machines. The main objective in these architectures is to minimize the execution time on the execution of applications. However, the energy consumption is becoming a problem, as computers grow in size.

The goal for the execution of simulations are maximum time performance and minimum energy consumption. The execution of a simulation can be done using different number of processors available in a multithreaded system. Executing with different number of processors can give different results both in terms of time and energy. In addition, it is not clear how many processors should be used to achieve the best performance per energy used.

In this work we present an empirical approach these problems. We perform an energy efficiency evaluation executing the protocol presented in (Zeigler 2017), in parallel computers, and we discuss a few results obtained when studying the benefits of executing it in a parallel computer empirically. To perform the evaluation, we use an implementation that uses shared-memory parallelism and the thread pool design pattern. It was built as a feature of Cadmium (Vicino et al. 2015), a DEVS simulator. We conducted empirical evaluations to compare execution times and energy consumption varying the number of threads used. Both synthetic and real-life models were used to conduct experimentation. This work is an extension of the one presented in (Lanuza et al. 2020).

2 BACKGROUND AND RELATED WORK

In this section we discuss the related concepts used on this work. First, we discuss the protocol used to perform simulations. Second, we show how this protocol was implemented. In third place, we describe a benchmark to generate different models. Finally, we describe the energy efficiency approach used to perform our evaluation.

2.1 Parallel DEVS Protocol

In discrete-event simulation, the operation of a system is represented as an ordered sequence of events, where each event occurs at an instant in time. There have been different research efforts related to running discrete event simulations in parallel. The reader can find the main efforts in the Proceedings of the PADS conferences between 1990 and 2019, and an introduction to this field can be found in (Fujimoto 1999).

The main methods presented in Parallel Discrete-Event Simulation (PDES) consist of the concept of Logical Processes (LPs). LPs function as the simulation entities, which do not share any state variables, and interact with each other through timestamped event messages. The major challenge in PDES is being able to produce the same results as in a sequential execution. Synchronization among these LPs is violated when one of the LPs receives an event that is older than the current clock time of the recipient LP. Such violation is referred to as causality error. To deal with causality errors, different synchronization techniques have been proposed (Fujimoto 1990; Fujimoto 1999).

In this work we focus on the parallel execution of DEVS simulations. In DEVS, atomic models are used to define behavior and coupled models are used to specify the structure of the system under study. DEVS is a mathematical formalism that provides a theoretical framework to think about modeling using a hierarchical, modular approach. This formalism is proven universal for discrete-event simulation modeling, meaning that any model described by other discrete-event model formalism has an equivalent model in DEVS. In DEVS there is clear separation between model and simulation: models are described using a formal notation, and simulation algorithms are defined for executing any model. Parallel DEVS (PDEVS) (Chow and Zeigler 1994; Chow et al. 1994) was introduced to deal with tie-breaking of simultaneous events and better handling in when simultaneous events occur.

To implement parallel DEVS simulations, there have been efforts on traditional optimistic (Liu and Wainer 2009; Nutaro 1999) as well as conservative approaches (Jafer and Wainer 2011). Nevertheless, as stated in (Zeigler 2017) only some conservative techniques have been proven to exactly represent the behavior of the DEVS reference simulator, for example the ones presented in (Adegoke et al. 2013; Cardoen et al. 2018). This feature distinguishes DEVS from the many other simulation engines derived from the LP approaches, which cannot reproduce the behavior of the DEVS reference simulator in all scenarios.

A different approach proposed in (Zeigler 2017) enables the execution of simultaneous events in parallel. The main idea behind this approach is to allow a simple and error free algorithm for the execution of DEVS simulations by identifying the tasks that are independent on the execution, and therefore can be executed in parallel guaranteeing a correct execution. The protocol consists in the following steps:

```
1.  Until a specified number of global transitions done
2.  Do global transition {
3.    For each imminent model (own tN=global tN), compute output and send it to receivers (*)
4.    For each active model (imminent and input receiver), compute state transition (internal, external,
confluent) (*)
5.    Send own tN
6.    Advance global clock, global tN = min active tNs }
```

Steps 1 and 2 set the main loop for the simulation: it will finish after a sequence of global transitions Each global transition is a sequence of tasks where all the events occurring on a specific time are computed, and the simulation time advances. In Step 3, the output function is calculated for every imminent model and the outputs are converted into inputs and sent to the receiver components. This loop can be executed in parallel (*). In step 4, the receivers and imminent components compute their state transitions. The imminent components compute an internal transition if they do not receive any input, and a confluent function when they do. Receivers compute their external transition. This step can also be executed in parallel (*). In Steps 5 and 6 each component sends the time for its next event and the minimum of then is calculated to advance the simulation time.

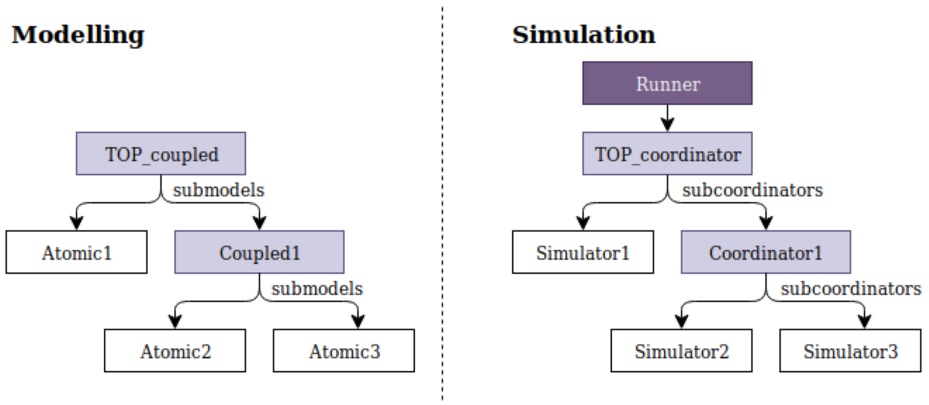

Figure 1: A DEVS model defined by a user and the corresponding simulation components.

The protocol guarantees the correct execution of the simulation in the sense that if the component models are DEVS models, then the result is also a well-defined DEVS coupled model. As it can be seen on the protocol, the execution time can be reduced by applying the parallelism on the execution of the simulation in two situations:

- When multiple atomic models have their internal transitions scheduled at the same time, the output functions can be executed in parallel; after that, all internal transitions can be executed in parallel.

- When the output of a model is connected to the input of one or more models, the internal transition of the first model can be computed concurrently with the external transitions of the receiving models.

In (Zeigler 2017) a theoretical analysis was made with this idea, concluding that speedups are related to specific model characteristics, such as the level of activity defined as the probability of being imminent on a global transition. The level of coupling is defined as the probability of a model of receiving an external event. The theoretical analysis concludes that this protocol can achieve an execution time 60% higher on average than the best possible parallel execution with this simple parallel implementation. Nevertheless, no results have been shown on how the simulations will perform applying this idea. We propose to give an empirical evaluation from an energy efficiency approach.

2.2 Cadmium Simulator

The software that we will use to perform our evaluation is the Cadmium DEVS simulator (Belloli et al. 2019), which is an open source simulator written in C++ originally designed to execute a single-threaded sequential algorithm described in (Vicino et al. 2017). Cadmium uses typed messages and typed ports, a time representation independent of the model implementation and it includes automated checking of some properties of the DEVS models for early error detection.

As in many DEVS simulation tools, Cadmium is developed in a way that the user only defines the models and does not need to know the simulation execution details. The user defines the coupled and atomic models and then executes the simulation. In the case of Cadmium, this is done invoking a *runner* class, which, when initialized, creates a simulator for each atomic model and a coordinator for each coupled model. A simulator manages the state of the atomic model over the execution and triggers the atomic model's functions and a coordinator handles intercommunication and synchronization between subcomponents (coordinators and simulators), as described in Figure 1.

The simulation algorithms were changed to run multiple threads and accelerate the execution. For every global transition, we can see that the output functions of all the imminent models are executed concurrently, and then the same happens for the transition functions of all the active models (imminent and/or input receivers). If the model has many components or many global transitions are executed, the number of tasks will grow. Also, as tasks are user-defined output or transition functions, they could have a short execution time. In order to avoid the overhead of creating and destroying threads for each task, the parallel implementation uses a thread pool (*basic_thread pool* provided by *Boost.Thread*, a popular library for C++). To integrate the thread pool into Cadmium's architecture, it was added as a member variable of the runner class, and when child coordinators are instantiated, they save a reference to that thread pool. When the coordinators execute the functions to collect the outputs and execute the transitions, they submit calls to the thread pool instead of executing it in the same thread.

2.3 DEVStone Benchmark

To perform our empirical evaluation, we used the DEVStone synthetic benchmark (Wainer et al. 2011). This benchmark creates varied synthetic models with different structure that are representative of those found in real world applications. Hence, it is possible to analyze the efficiency of a simulation engine with relation to the characteristics of a category of models of interest. A DEVStone generator builds varied models using the following parameters:

- type: different structure and interconnection schemes between the components.
- depth: the number of levels in the modeling hierarchy.
- width: the number of components in each intermediate coupled model.
- internal transition time: the execution time spent by internal transition functions.
- external transition time: the execution time spent by external transition functions.

Internal and external transition functions are programmed to execute an amount of time specified by the user, which execute Dhrystones (Weicker 1984). The Dhrystone synthetic benchmark, intended to be representative for system programming, uses published statistics on the use of programming language features, and it is available for different programming languages (C++, Java, Python, etc.). Dhrystone code consists of a mix of instructions using integer arithmetic; therefore, it is an appropriate choice for analyzing models like DEVS in which we use discrete state variables. Any simulation tool based on a programming language in which Dhrystones can be defined and executed, can be adapted to execute the DEVStone benchmark.

DEVStone uses two key parameters: d, the depth and w, the width, with which a DEVStone model of any given size can be implemented, where each depth level, except the last, will have w-1 atomic models, and each atomic model provides customizable Dhrystone running time. The inner model of such scheme is made up of a coupled model that embeds a single atomic model. In general, being d the depth and w the width, we build a coupled model with d coupled components in the hierarchy, all of which consist of w-1 atomic models (in the lower level of the hierarchy, the coupled component consists of a single atomic model). The model can be conceived as a coupled component that wraps w atomic components and another coupled component, which in turn has a similar structure. The connection with the exterior is done by one input and one output links. The input feeds the first coupled component; the coupled component then builds links from the single input each of its subcomponents. Non-hierarchical modeling and simulation tools can use DEVStone by defining d=0 and use a single-level model to evaluate the performance. DEVStone uses four several types of internal and external structures:

- LI: models with a low level of interconnections for each coupled model. Each coupled component has only one input and one output port. The input port is connected to each component but only one component produced an output through the output port.
- HI: models with a high level of input couplings. HI models have the same number of atomic components with more interconnections: each atomic component (a) connects its output port to the input port of the (a+1)th component.
- HO: models with high level of coupling and numerous outputs. The HO type models have a more complex interconnection scheme with the same number of atomic and coupled components. HO coupled models have two input and two output ports in each level. The second input port in the coupled component is connected to its first atomic component.
- HOmod: models with an exponential level of coupling and outputs. The HOmod models increment the message traffic, and they exponentially explode the interchange of messages among coupled models. They use a second set of (w-1) models where each one of the atomic components triggers the entire first set of (w-1) atomic models. These in turn have their outputs connected to the second input of the coupled model within the level. With such interconnections, the inner model receives several events that has an exponential relationship between the width and the depth at each level. External events are forwarded by each coupled component to its w-1 atomic children and to its coupled child, and the process is repeated in each coupled module until the arrival to the leaf component.

2.4 Energy Efficiency

As mentioned in previous sections, our main concern in this work is to evaluate the energy efficiency on the execution of DEVS simulations. To achieve this, we propose in our approach to measure two parameters for every execution of a simulation with different number of processors: execution time (in seconds) and energy used to execute the simulation (in Joules).

To measure execution time there are several libraries available, some of them as tools implemented in operating systems. However, to measure energy consumption in the execution of programs, only some exist and very few have been proven to perform accurate measures.

```
Energy measurements:
PACKAGE_ENERGY:PACKAGE0   176.450363J (42.9W)
PACKAGE_ENERGY:PACKAGE1    75.812454J (18.4W)
DRAM_ENERGY:PACKAGE0       11.450363J (2.9W)
DRAM_ENERGY:PACKAGE1        8.450363J (2.0W)
PP0_ENERGY:PACKAGE0       118.029236J (28.7W)
PP0_ENERGY:PACKAGE1        16.759064J (4.1W)
```

Figure 2: An example of RAPL measurements.

In this work we use the tool Running Average Power Limit (RAPL) (David et al. 2010; Hähnel et al. 2012;). RAPL is a tool designed by Intel that allows to monitor energy consumption across different domains of the CPU chip, attached DRAM, and on-chip GPU with promising accuracy (Khan et al. 2018). RAPL is included in the papi library (Terpstra et al. 2010). An example of the execution of the RAPL component can be seen in Figure 2. Here we can see how different values can be obtained. For each multiprocessor the tool reports a PACKAGE value, in this example packages 0 and 1. For each package, the energy is measure for the processors (PP_ENERGY), main memory (DRAM_ENERGY) and the multicore and main memory combined (PACKAGE_ENERGY). In this work we use as the measurement for our evaluation the PACKAGE_ENERGY, as it provides the most complete information about the energy used on an application.

The energy efficiency of a program is defined as performance (i.e. floating-point operations (flops)/seconds) per energy unit (i.e Watts per second) (Rauber at al. 2017). To measure the energy efficiency, we will use a metric called Energy-Delay Product (EDP) metric (Horowitz et al. 1994; Rong et al. 2010):

$$EDP(p) = E(p) * T(p) \tag{1}$$

Where *p* is the number of processors, *E(p)* is the energy consumed by executing with *p* processors and *T(p)* is the execution time for the application using *p* processors. When comparing different EDP values, smaller EPD indicates a better energy efficiency, a larger performance per energy unit.

3 EXPERIMENTAL SETUP

To perform our energy efficiency analysis, we used a desktop machine with an intel i5-9400 multiprocessor, with 6 cores and 16GB of RAM, in Figure 3 a graphical representation for the processor's architecture is show. Figure 3 was made with the hwloc tool [23] (Broquedis et al. 2010).

Our approach is to evaluate the execution of different problems on this platform by considering different number of threads on the thread pool for the execution of the Cadmium simulator. In addition, to evaluate the simulation executions on this platform, we use two types of DEVS models: synthetic models and a real-world model.

In first place, we performed experiments using the DEVStone benchmark. A version of the DEVStone benchmark was used to execute in the Cadmium simulator. This way, DEVStone models were generated automatically. As detailed on Section 2.3, DEVStone allows to generate synthetic models that represent the structure of real-world models. The structure chosen for this evaluation, has 5 levels of depth, and a width of 100, to represent a complex model. The time for the internal and external transitions we chose is 100 ms. In addition, to evaluate different degrees of complexity on the models, we use all possible types of models that can be generated with DEVStone. Therefore, we consider four models, one for each type: LI, HI, HO and HOmod with the following parameters:

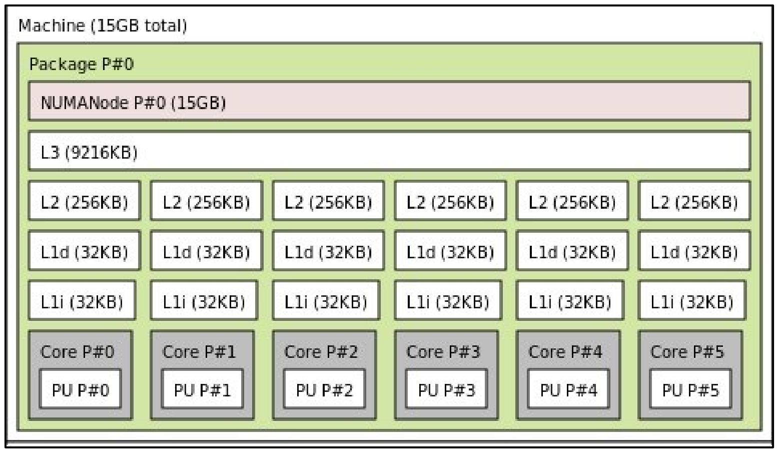

Figure 3: Desktop platform with Intel i5-9400 multicore processor.

- width = 100.
- depth = 5.
- internal transition time = 100 ms.
- external transition time = 100 ms.

These 4 models have the same amount of atomic and coupled models, the difference is in the connections they have. As the connections increase between the subcomponents more computation is needed to execute them and therefore more time and energy is required to execute.

In second place, we perform our evaluation on a complex real-world problem: the study of metabolic pathways in biological cells. In biology and biochemistry, biological cells and their subsystems play a fundamental role and their study is essential. One of the main phenomena occurring in cells are called “metabolic pathways", a series of linked reactions that produce transformations of different metabolites (metabolites are small molecules, like hydrogen or oxygen, that are either consumed or produced by chemical reactions in the cell). These reactions conform the metabolism of the cells and their life; therefore, the study of this phenomenon has become popular in different fields related to medicine, biology, and healthcare, among others.

The model used in this work is the metabolic pathways for the E. Coli bacteria. This model involves all the information regarding the cell structure, enzymes and reactions involved in the E. Coli metabolic pathways. The DEVS model structure for this problem has 1608 atomic models and 23 coupled models distributed in 4 levels of hierarchy. This model is defined by stochastic behavior. More information about this problem and the complete model can be found in (Belloli et al. 2016; Belloli 2019).

4 EXPERIMENTAL RESULTS

In this section we present the results obtained when we executed the models described in the previous section on the multicore platform discussed in Section 3. The results from the experiments are show on Tables 1, 2, 3, 4 and 5. As expected. the DEVStone model that requires less time and energy to execute from all is the LI, followed by the HI, HO and HOmod in increasing model complexity order.

On Table 1 the results for the LI model are presented. For this model, the best result in time is obtained with 6 thread and the best result in energy is obtained by the sequential version. The minimum EDP is obtained by using 6 threads, and therefore the best energy efficiency. The speedup obtained with 6 threads is 1.6 and the energy consumed is increased 15.55% with respect of the sequential version. It is important to notice that using more threads is not always better, as using 2, 3 and 4 threads is worse in time and energy than the sequential version.

Table 1: LI model execution results

Number of Threads	Time (seconds)	Energy (Joules)	EDP
1	1.011	21.795	22.041
2	1.817	35.999	65.399
3	1.250	30.260	37.825
4	1.016	29.170	29.622
5	0.798	27.655	22.078
6	0.622	25.185	15.665

Table 2: HI model execution results

Number of Threads	Time (seconds)	Energy (Joules)	EDP
1	10.198	218.28	2226.061
2	18.304	363.59	6655.260
3	13.074	320.98	4196.492
4	9.895	291.09	2880.527
5	6.923	262.86	1819.990
6	6.172	255.06	1574.464

On Table 2 the results for the HI model are presented. The best result in time is obtained by using 6 threads. The speedup obtained with 6 threads is 1.65. To obtain this performance the configuration uses 16.84% more energy than the sequential version, which achieves the minimum energy consumption. Nevertheless, evaluating the energy efficiency in terms of the EDP metric, the most energy efficient execution is obtained by using 6 threads, as this configuration is the best in terms of performance per energy used.

Table 3: HO model execution results

Number of Threads	Time (seconds)	Energy (Joules)	EDP
1	22.613	472.16	10680.019
2	31.980	624.81	19981.995
3	26.527	583.75	15485.452
4	22.998	591.12	13594.742
5	19.904	571.07	11367.017
6	19.173	566.67	10864.947

The results for HO model are show in Table 3. As in the previous cases, the best results are obtained by using 6 threads in terms of time and by the sequential version in terms of energy. The speedup obtained with 6 threads is 1.17 and the energy used is 16.67% higher than the sequential version. In this case, the lower EPD is obtained by the sequential version. The difference between the 6 thread and the sequential execution is 1.7% in EDP.

In Table 4 the results for the HOmod model are presented. Again, the best result in time is obtained by using 6 threads. The speedup obtained with 6 threads is 1.74. As in previous results, to obtain this performance, the configuration uses 15.14% more energy than the sequential version, which achieves the minimum energy consumption. But. evaluating the energy efficiency in terms of the EDP metric, the most

Table 4. HOmod model execution results

Number of Threads	Time (seconds)	Energy (Joules)	EDP
1	468.807	9960.56	4669578.334
2	909.233	15206.47	13826227.700
3	657.079	15595.63	10247559.850
4	479.343	14138.08	6776988.373
5	308.811	12049.02	3720870.996
6	268.409	11738.41	3150694.293

energy efficient execution is obtained by using 6 threads, as this configuration is the best in terms of performance per energy used.

Table 5. Metabolic pathways model execution results.

Number of Threads	Time (seconds)	Energy (Joules)	EDP
1	116.081	2097.18	243442.382
2	83.241	1925.06	160245.074
3	63.113	1851.83	116875.102
4	61.464	1930.13	118634.282
5	60.751	1946.77	118269.358
6	59.5151	1951.31	116132.409

Finally, the results for the metabolic pathways are presented in Table 5. Here, the best result in time and energy is obtained by using 6 threads, and therefore the minimum EDP is obtained by this configuration. The speedup obtained with 6 threads is 1.95 and the energy reduction is 7,47% compared to the sequential version.

These results show how using the same amount of threads as processors on the multicore, the best result is obtained in energy efficiency in almost all cases. Even in the cases when the sequential version obtains the minor energy consumption, the best performance per energy used is obtained by using 6 threads. Another important insight from these results is that in many cases using 2, 3 and 4 processors is worst in time and energy than the sequential version.

5 CONCLUSIONS AND FUTURE WORK

In this work we presented an empirical evaluation of the energy efficiency obtained by executing DEVS simulations in parallel in multithreading architectures. The results obtained show how parallel executions can achieve better performance than sequential version in terms of performance and energy consumption. Nevertheless, still more work is required to achieve more efficient simulations in time and energy. As future work we perform more implementation on different multithreading architectures and to compare the thread pool structure against other patterns for parallel programming, as the fork-join approach provided by the OpenMP library.

REFERENCES

Adegoke, A., H. Togo, and M.K. Traoré. 2013. "A Unifying Framework for Specifying DEVS Parallel and Distributed Simulation Architectures," *SIMULATION* 89(11):1293–1309.

Belloli, L. 2019. "Defining DEVS Models for Biology Pathways in E-Coli Cells", Master Thesis from Universidad de Buenos

Aires, Argentina.

Belloli, L., D. Vicino, C. Ruiz-Martin and G. Wainer. 2019. “Building DEVS Models with the Cadmium Tool”. In *Proceedings of the 2019 Winter Simulation Conference*, edited by N. Mustafee, K.-H.G. Bae, S. Lazarova-Molnar, M. Rabe, C. Szabo, P. Haas, and Y.-J. Son, 45–59. Piscataway, New Jersey, USA: Institute of Electrical and Electronics Engineers, Inc.

Belloli, L., G. Wainer, and R. Najmanovich. 2016. "Parsing and Model Generation for Biological Processes". in *Proceedings of the Symposium on Theory of Modeling and Simulation (TMS-DEVS),* edited by F. Barros, X. Hu, H. Prahofer and J. Denil. Art. 21. Pasadena, CA, USA, Institute of Electrical and Electronics Engineers, Inc.

Broquedis, F., J. Clet-Ortega , S. Moreaud, N. Furmento, B. Goglin, G. Mercier, S. Thibault, and R. Namyst. 2010. “hwloc: A Generic Framework for Managing Hardware Affinities in HPC Applications”. In *Proceedings of the 18th Euromicro Conference on Parallel, Distributed and Network-based Processing (PDP)* edited by M. Danelutto, J. Bourgeois and T. Gross. 180-186. Washington DC, USA: Institute of Electrical and Electronics Engineers Computer Society.

Cardoen, B., S. Manhaeve, Y. Van Tendeloo, Y., and J. Broeckhove. 2018. “A PDEVS Simulator Supporting Multiple Synchronization Protocols: Implementation and Performance Analysis”. *SIMULATION* 94(4):281–300.

Chow, A. and B. P. Zeigler. 1994. “Parallel DEVS: a Parallel, Hierarchical, Modular, Modeling Formalism”. In *Proceedings of the 1994 Winter Simulation Conference*, edited by J. D. Tew, S. Manivannan, D. A. Sadowski, and A. F. Seila, pp. 716–722. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Chow, A.C., B. P. Zeigler and D. H. Kim. 1994. "Abstract Simulator for the Parallel DEVS Formalism". In *Proceedings of the Fifth Conference on AI, Simulation, and Planning in High Autonomy Systems*, edited by Paul. A. Fishwick. pp. 157-163. Gainesville, FL, USA: Institute of Electrical and Electronics Engineers, Inc.

David, H., E. Gorbatov, U. R. Hanebutte, R. Khanna and C. Le. 2010. "RAPL: Memory Power Estimation and Capping". In *Proceedings of ACM/IEEE International Symposium on Low-Power Electronics and Design (ISLPED)*, edited by: V. Oklobdzija, B. Pangle, N. Chang, N. Shanbhag, and C. H. Kim, 189-194. New York, NY, USA: Association for Computer Machinery.

Fujimoto, R, M. 1990. Parallel Discrete Event Simulation, *Communications of the ACM* 33(10), 30–53.

Fujimoto, R. M. 1999. Parallel and Distribution Simulation Systems. 1st. ed. New York, NY, USA: John Wiley & Sons, Inc.

Hähnel, M., B. Döbel, M. Völp and H. Härtig. 2012. “Measuring Energy Consumption for Short Code Paths using RAPL”. *ACM SIGMETRICS Performance Evaluation Review* 40(3):13-17.

Horowitz, M., T. Indermaur, and R. Gonzalez. 1994. “Low-Power Digital Design”. In *Proceedings of 1994 IEEE Symposium on Low Power Electronics*, edited by John H. Wuorinen. 8–11. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Jafer, S. and G. Wainer. 2011. “Conservative Synchronization Methods for Parallel DEVS and Cell-DEVS”. In *Proceedings of the 2011 Summer Computer Simulation Conference (SCSC ’11),* edited by P. Kropf, A. Abhari, M. K. Traoré and H. Vakilzadian .60-67. Vista, CA: Society for Modeling & Simulation International.

Khan K. N., M. Hirki, T. Niemi, J. K. Nurminen, and Z. Ou. 2018. RAPL in Action: Experiences in Using RAPL for Power Measurements. *ACM Trans. Model. Perform. Eval. Comput. Syst*. 3(2): Art. 9.

Lanuza, J., G. Trabes, and G. Wainer. 2020. “Parallel Execution of DEVS in Shared-Memory Multicore Architectures”. In *Proceedings of the 2020 Spring Simulation Conference,* edited by F. Barros, X. Hu, H. Kavak, and A. del Barrio. 1-11. San Diego, CA, USA, Society of Computer Simulation International.

Liu, Q. and G. Wainer. 2009. "A Performance Evaluation of the Lightweight Time Warp Protocol in Optimistic Parallel Simulation of DEVS-Based Environmental Models". ACM/IEEE/SCS 23rd Workshop on Principles of Advanced and Distributed Simulation, Lake Placid, NY, USA. pp. 27-34.

Nutaro, J. 2009. “On Constructing Optimistic Simulation Algorithms for the Discrete Event System Specification”. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*19(1):1-21.

Rauber T., G. Rünger and M. Stachowski. 2017. "Towards New Metrics for Appraising Performance and Energy Efficiency of Parallel Scientific Programs". 2017. In *Proceedings of the IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData),* edited by Y. Wu, G. Min, N. Georgalas, A. Al-Dubi, X. Jin, L. Yang, J. Ma and P. Yang. pp. 466-474. Exeter, UK: Institute of Electrical and Electronics Engineers, Inc.

Rong. G., F. Xizhou, S. Shuaiwen, C. Hung-Ching, L. Dong, and K.W. Cameron. 2010. “PowerPack: Energy Profiling and Analysis of High-Performance Systems and Applications”. *IEEE Transactions on Parallel and Distributed Systems* 21(5):658 –671.

Terpstra D., H. Jagode, H. You and J. Dongarra. 2010. “Collecting Performance Data with PAPI-C”. 2010. In P*roceedings of the 3rd International Workshop on Parallel Tools for High Perfoemance Computing,* edited by M. S. Müller, M. M. Resch, A. Schulz, and W. E. Nagel. 157-173., Dresden, Germany: Springer Berlin Heidelberg.

Vicino, D., D. Niyonkuru, G. Wainer, and O. Dalle. 2015. “Sequential PDEVS Architecture”. In *Proceedings of the Symposium on Theory of Modeling & Simulation: DEVS Integrative M&S Symposium (DEVS ’15)*, edited by F. Barros, M. H. Wang, H. Prähofer, and X. Hu. 165-172, San Diego, CA, USA: Society for Computer Simulation International.

Wainer, G., E. Glinsky and M. Gutierrez-Alcaraz. 2011 "Studying Performance of DEVS Modeling and Simulation Environments using the DEVStone Benchmark," *SIMULATION* 87(7):555-580.

Weicker, R. P. 1984. “Dhrystone: a synthetic systems programming benchmark”. *Communications of the.ACM* 27(10):1013-1030.

Zeigler, B., H. Praehofer and T. G. Kim. 2000. *Theory of Modeling and Simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems*, San Diego, CA: Academic Press.
Zeigler, B., 2017. "Using the Parallel DEVS Protocol for General Robust Simulation with Near Optimal Performance". *Computing in Science & Engineering* 19(03):68-77.

AUTHOR BIOGRAPHIES

GUILLERMO G. TRABES is a Ph.D. student in Electrical and Computer Engineering (Carleton University) and Computer Science (Universidad Nacional de San Luis). His email address is guillermotrabes@sce.carleton.ca.

VERONICA GIL COSTA received her MSc (2006) and PhD (2009) in Computer Science, both from Universidad Nacional de San Luis (UNSL), Argentina. She is a former researcher at Yahoo! Labs Santiago hosted by the University of Chile. She is currently an associate professor at the University of San Luis and researcher at the National Research Council (CONICET) of Argentina. Her email address is gvcosta@unsl.edu.ar.

GABRIEL A. WAINER is Professor at the Department of Systems and Computer Engineering at Carleton University. He is a Fellow of the Society for Modeling and Simulation International (SCS). His email address is gwainer@sce.carleton.ca.