Exploring Scalability in System-Level Simulation Environments for Distributed Cyber-Physical Systems

Herman Kelder
University of Amsterdam (UvA), The Netherlands
Vrije Universiteit Amsterdam (VU), The Netherlands
herman.kelder@student.uva.nl

ABSTRACT

Industrial Cyber-Physical Systems (CPS) are sophisticated interconnected systems that combine physical and software components driving various industry sectors worldwide. Distributed CPS (dCPS) consist of many multi-core systems connected via complicated networks. During the development of dCPS, researchers and industrial designers need to consider various design options which have the potential to impact the system's behaviour, cost, and performance. The resulting ramification in size and complexity poses new challenges for manufacturing companies in designing their next-generation machines. However, objectively evaluating these machines' vast number of potential arrangements can be resource-intensive. One potential alternative is to use simulations to model the systems and provide initial analysis with reduced overhead and costs.

This literature review investigates state-of-the-art scalability techniques for system-level simulation environments, i.e. Simulation Campaigns, Parallel Discrete Event Simulations (PDES), and Hardware Accelerators. The goal is to address the challenge of scalable Design Space Exploration (DSE) for dCPS, discussing such approaches' characteristics, applications, advantages, and limitations. The conclusion recommends starting with simulation campaigns as those provide increased throughput, adapt to the number of tasks and resources, and are already implemented by many state-of-the-art simulators. Nevertheless, further research has to be conducted to define, implement, and test a sophisticated general workflow addressing the diverse sub-challenges of scaling system-level simulation environments for the exploration of industrial-size distributed Cyber-Physical Systems.

KEYWORDS

Scalable Simulation Environments, System-Level Simulation, Distributed Cyber-Physical Systems, Design Space Exploration, Simulation Campaigns, Parallel Discrete Event Simulations

1 INTRODUCTION

Nowadays, various information-technology sectors worldwide are driven by complex interconnected Cyber-Physical Systems (CPS), which deeply intertwine physical and software components. These systems support innovation and research in a variety of crucial industrial sectors, like health industries, industrial automation, robotics, avionics and space. Usually, the compute infrastructure of such CPS consists of many heterogeneous multi-core or many-core systems, connected via complex networks to create a *distributed Cyber-Physical Systems* (dCPS) [1]. These subsystems implement a variety of functionalities (e.g., monitoring and control) to provide specific services at the system-level. The complexity in the infrastructure of dCPS creates challenges for manufacturing companies, such as ASML, Canon Production Printing, and Philips, in designing their next-generation lithography scanner machines, industrial printers, and X-ray machines, respectively [37].

During the development of such systems, researchers and industry designers often encounter "What-if" questions (i.e., "What happens if we add more communication channels between subsystems?", "What if we add hardware accelerators to subsystems?", "What if we merge subsystems?"). Decisions on those design questions have the potential to impact a large amount of the system's behaviour, cost, energy consumption, and performance. The vast number of possible configurations based on these decisions results in a significant increase in resources required in the design process. Evaluating design options early on, based on design constraints, Key Performance Indicators (KPIs), and other metrics could significantly reduce this effect. Building a prototype to answer the questions usually is cost-intensive, depending on the system to be implemented. One potential alternative is to model such systems to provide an initial analysis with reduced overhead and costs. There are various ways to accomplish that, like simulations, analytical models, or other performance estimations.

Nevertheless, large sets of configurations that need to be evaluated provide a significant challenge. A property of a design evaluation environment that could address the increased workload is *scalability*. This literature study looks into different scalability techniques that can be applied to the evaluation of dCPS designs. In Section 2, the basic concepts are introduced and the context of this literature study is addressed. In Section 3, an overview of the related works is discussed, focusing on different methodologies to achieve scalability. In Section 4, a discussion on the methodologies and their applications is presented. Finally, a conclusion on the literature study and its findings is given in Section 5.

2 BACKGROUND

In the background section, basic concepts are introduced and the context of this literature study is addressed. First, the DSE2.0 project [22] on the challenges of *Design Space Exploration* (DSE) for distributed Cyber-Physical Systems is highlighted, as it is the context in which this study is performed. After DSE2.0, the definition of scalability is examined, which is the goal of the research that will be discussed. Finally, the scope of the study is briefly considered.

2.1 Design Space Exploration (DSE)

One potential approach for providing designers with early directions in their design process is Design Space Exploration [22]. DSE is the process of traversing a solution space consisting of potential

design solutions (*design points*). The goal is to discover solutions that best satisfy a set of *design objectives*, which describe the functional and non-functional requirements of a system. Functional requirements define system features or functionalities which determine how the system behaves and operates. The non-functional requirements describe constraints or restrictions on system attributes in areas such as security, reliability, performance, maintainability, scalability, and usability. Common examples of design objectives include energy consumption, cost, or throughput. The number of available design choices determines the complexity and size of the design space. DSE is classified as either a single or multi-objective optimisation problem, which can be subject to design constraints that capture limitations and requirements imposed on the design points. DSE has already been adopted by many areas in the computer systems field, such as low-level hardware design for Systems-on-a-Chip (SoC) [42] and Multiprocessor System-on-a-Chip (MPSoC) [30], or the co-design of hardware and software [53].

Nevertheless, applying it to complex distributed Cyber-Physical Systems is a relatively uncharted area [47]. One ongoing research project in cooperation with ASML is "DSE2.0" [22]. The authors consider the entirety of the state-of-the-art DSE process under scrutiny for advancing from, e.g. embedded systems to large industrial machines like the ASML TwinScan machines. Their general workflow, illustrated in Figure 1, outlines the four main stages within the DSE framework: (i) Models, (ii) Design Space, (iii) Exploration, and (iv) Results. As it is a general workflow, the implementation of the steps will need to be reconsidered to achieve their respective objectives based on the underlying system and design choices to be explored.

In the first stage, *Modelling*, the modelling stage, the system artefacts (software, design choices, platforms, etc.) are discovered, described, and abstracted. These models are then mapped into a condensed system representation, creating one model combining software and hardware. This initial model is most often based on an existing system and can be used as a baseline for further steps of the DSE workflow.

Using this model, a *Design Space* is spanned in the second stage based on the available design choices. The initial design space encapsulates all possible design points of the model. Nevertheless, a preliminary pruning phase can be performed based on external constraints or incompatibilities (i.e., two hardware components are incompatible with each other).

Exploration of the design space and capturing the results of system configuration evaluations is the third stage of their general DSE workflow. A search algorithm is applied in this stage to determine which design points should be evaluated. Evaluating a design point is defined as the study of its extra-functional behaviour, and the capture of measurable Key Performance Indicators [22]. The search algorithm can dynamically prune design points based on evaluation results, potentially decreasing the size of the design space.

The final stage of the general DSE workflow are intermediate outcomes or conclusive recommendations. Intermediate data can be used to validate, tune, adjust, or change the search algorithm and models used throughout the DSE workflow. One example is a multi-level hierarchical search approach whereby the abstraction level can be adjusted (i.e., decreasing the abstraction level in multiple runs).

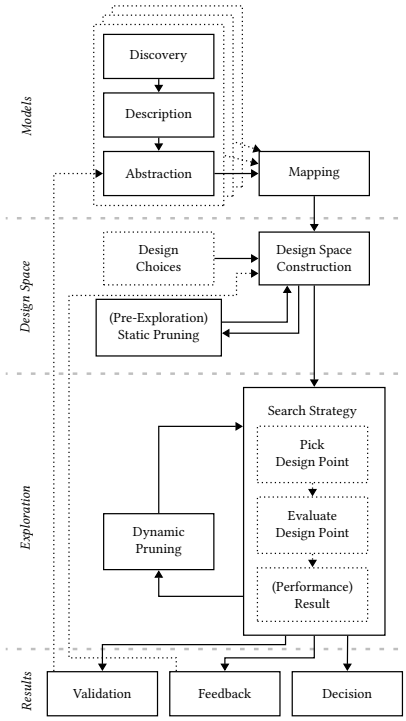

Figure 1: General DSE workflow by Herget et al. [22].

At the end of the process, the best design decisions are presented to the designer to guide decision-making [22].

Nevertheless, due to the size and complexity of dCPS, various challenges arise in the DSE process. In their position paper in 2022 [22], the authors identified two in specific: (i) Modelling complex dCPS, and (ii) Scalable DSE. The first challenge concerns the first stage, *Models*, of the general workflow. Models of the heterogeneous subsystems in the dCPS need to be created. However, in comparison to less complex systems like SoC and MPSoC, this is a huge challenge [22]. Manual generation of those models in a timely manner is therefore deemed infeasible. This challenge of modelling such a system requires a different methodology. Herget et al. [22] propose (semi-)automatic model interference of the application and platform model as a viable option for DSE of dCPS.

The second challenge, considering the scalability of simulation environments for dCPS, is the point of interest in this literature study. Complexity and heterogeneity of dCPS increase the number of design choices, resulting in a vast expansion of the design space. This is further exacerbated by conditions such as dynamic behaviour of the system workload settings, and more. An efficient approach to DSE is required, which can utilise scalability of the simulation environment or of search and pruning strategies. This challenge covers the *Exploration* stage of the general workflow, where it can especially be applied to the Evaluation Design Point and (Performance) Result section steps.

2.2 Scalability

One potential approach for performance improvement in the DSE process is the scalability of the simulation environment. While

the term scalability is widely used in modern computer science, a precise definition is still highly debated [36] [23] [33].

A rather colloquial usage of the term "scalability" refers to the ability of a system, network, or process to handle a growing amount of work, or its potential to be enlarged in order to accommodate that growth. In the context of simulation, this can be expressed as the ability to handle a growing number of simulation entities (Size capability), increase the performance of a single simulation run (Time capability), or handle more extensive and complex simulations (Complexity capability) [33]. This ability is most often achieved by improvements to the hardware design of the system running the simulations. Law [33] describes this factor as the "architectural capability" of a system and hence defines scalability as a ratio between the simulation and architectural capability. Another - more precise - definition has been presented by Weinstock and Goodenough: "Scalability is the ability to handle increased workloads by repeatedly applying a cost-effective strategy for extending a system's capacity." [54].

Despite the lack of a universally accepted definition, the general interpretation within the literature has similar aspects: to improve the capability of a system through hardware, software, or a combination of both. Hence, a wide range of approaches has been introduced, ranging from specialised hardware [56] to independent replications [49] and decomposition into logical processes [15].

Scalability capabilities can be addressed and implemented in different ways by a simulation environment, which is one technique to capture and evaluate a design point's behaviour. Simulations mimic a given system model, resulting in the opportunity to evaluate its performance based on the defined design objectives. In order to make the evaluation suitable for efficient, time-constrained analysis, the model's degree of granularity can be adjusted. The level of abstraction is a trade-off between evaluation speed and accuracy. A high level provides an increased evaluation speed (Time capability), but details of the system are lost (Complexity capability), often resulting in a lower accuracy [22]. Different simulation frameworks and techniques make use of this modifier depending on their intended application. The highest granularity is register-transfer level simulation, modelling the system's digital signals between registers and combinational logic. Higher levels of abstraction include, amongst others, cycle-accurate [45] [6], transaction-level [7], and trace-driven [48].

2.3 Scope

This literature study is based on the Design Space Exploration (DSE) of distributed Cyber-Physical Systems (dCPS). Specifically, the literature study is set in the scope of the DSE2.0 project [22] and focuses on challenge (ii) "Scalable DSE". The core area of interest will be scalability approaches that are applicable to the system-level simulation environments. Due to the complex, heterogeneous nature of dCPS, low abstraction levels for the simulation environment would be too computationally intensive and is therefore not of interest. Hence, approaches such as cycle-accurate and register transaction-level are not considered. Instead, the related works section will discuss some of the techniques proposed in the literature which can operate at a higher abstraction level and could potentially be applied to scale a simulation environment in the context of DSE for a dCPS.

3 RELATED WORK

The goal of this section is to present the state-of-the-art on methods and techniques for the scalability of system-level simulation environments. There are various concepts under investigation: (i) Campaigns, Section 3.1, where multiple instances of the simulation environment are executed at the same time, (ii) Parallel Discrete Event Simulation, Section 3.2, where the model is split into separate processes, and (iii) Hardware Acceleration, Section 3.3, which makes use of specialised hardware in the simulation environment.

3.1 Simulation Campaigns

The first methodology in scaling system-level simulation environments is the concept of *simulation campaigns* (or *Multiple Replications In Parallel (MRIP)*) [38] [21] [39]. Instead of running a single instance of the simulation environment with a single configuration on a single resource (i.e., completely sequential), multiple instances of the simulation environment with different (or the same) configurations are run on separate resources in parallel. This provides the opportunity to evaluate multiple design points at the same time.

Simulation campaigns can be configured in various ways, providing flexibility and adaptivity in the simulation strategy. This makes the technique applicable to a variety of different simulation environments and fields of research, like stochastic simulations [11] or Markov chain simulations [49]. The freedom in the implementation of simulation campaigns has resulted in proposals of various approaches, including independent replicated execution [49], cloning [25], and more.

In order to apply simulation campaigns to a system, *problem decomposition* can be applied to split a complex problem into parts that are easier to conceive, understand, program, and maintain. In the context of DSE, the problem to which the decomposition is applied would be the evaluation of a set of design points generated by the exploration stage strategy. Problem decomposition defines the approach whereby an application is formulated into constituent processes. Two well-known problem decompositions are *task parallelism* and *data parallelism*. A concrete application of both approaches on a JPEG decoder is displayed by Hansson, Akesson, and Van Meerbergen [20]. Task parallelism focuses on separate processes or threads of execution, where the processes are behaviourally distinct. As an example of task parallelism, the JPEG decoder's three stages (i) Variable Length Decoding (VLD), (ii) Inverse Discrete Cosine Transformation (IDCT), and (iii) Colour Conversion (CC) can be mapped to separate resources, effectively creating a pipeline.

Instead of decomposing the application itself, data parallelism maps a set of tasks on disjoint parts of the data set. As an example of data parallelism, the image can be partitioned and processed on separate resources by replications of a slightly adapted version of the original JPEG decoder application. Additionally, the data parallelism decomposition will be used as it fits the representation of executing multiple simulation environments to evaluate different design points.

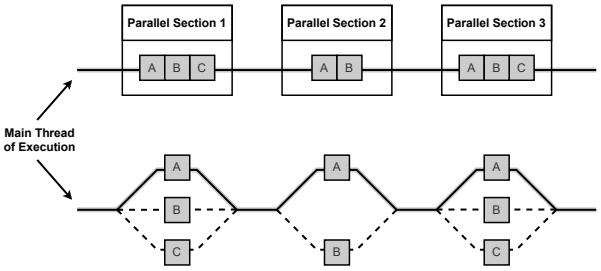

Figure 2: Illustration of the Fork/Join concept. A main thread of execution forks into multiple threads, which later join back into the main thread.

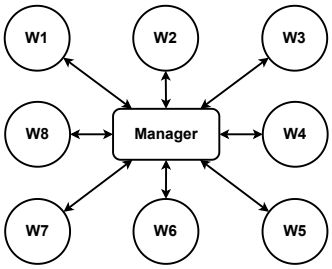

Figure 3: Illustration of the Manager/Worker concept. A manager process orchestrates a set of worker processes.

An overarching *execution model* needs to be defined to manage the decomposition and execution of the application (i.e., the simulation campaign). Amongst the various execution models are the fork-join model [32] and manager/worker model [46]. The fork-join model, illustrated in Figure 2, concerns an application where execution branches off in concurrent subprocesses at designated points. At a subsequent predefined point, execution "joins" (the branches return to the main application) and sequential operation resumes. When fork-join operations are nested recursively, the fork-join model becomes a parallel version of the divide-and-conquer model [34]. In the divide-and-conquer model, an application is recursively split into two or more sub-problems. This process continues until the sub-problems are simple enough to be solved directly.

The manager/worker model, in contrast to fork-join, assigns one process to be the manager and all other processes become workers. Figure 3 illustrates this concept. In this model, the manager process orchestrates the execution of the operation and worker processes. Generally, the manager process oversees, among other things, data distribution and aggregation, synchronisation, and communication. Workers can be identical or distinct processes executing concurrently.

Both models can be extended and adapted in many ways. For example, the manager/worker model could have a dynamic pool of processes (the number of available processes might increase or decrease) or a voting mechanism to assign a new manager in case the current manager process fails. The flexibility within the models allows for different approaches in the adoption and implementation of simulation campaigns in computer science field and other fields.

Many simulation environments have adopted the concept of simulation campaigns. A popular one that provides the opportunity to make use of this approach is the Omnet++ simulator [52]. Omnet++ has the built-in functionality to perform parameter studies and distributed stochastic simulations. Parameter studies are provided with a tool to group simulation runs (system configurations) into batches, where each batch executes its set of simulations sequentially inside a single process. Batches are scheduled for running so that they keep all CPUs busy. Parameters, such as batch size and number of CPUs, can be configured manually.

Another well-known simulation tool that makes use of simulation campaigns is Akaroa [11], which is also available within Omnet++. The tool focuses on quantitative stochastic simulation and speeding up the simulation process using Multiple Replications In Parallel (MRIP). In the domain of quantitative stochastic simulation, the runtime of simulations can depend on the initial values of the model. Those initial values dictate the results of the initial transient phase (i.e., the system has not reached a steady state), which should not be included in the data. Additionally, the runtime is also dependent on the convergence of the system (when has it stabilised or reached the desired sample size). Akaroa accumulates the data in a central process. Based on a predefined confidence/precision level, it determines whether more observations are required. The simulation campaign is terminated when it has accumulated enough observations.

3.2 Parallel Discrete Event Simulation (PDES)

The second methodology is *Parallel Discrete Event Simulation* (PDES). While simulation campaigns execute multiple instances of the simulation environment with different configurations in parallel, PDES executes a single simulation environment with a single configuration in parallel.

PDES builds on the concept of *discrete event simulation*. Discrete event simulation is an *event-driven* model of a system's behaviour, which assumes the system only changes state at discrete points in simulated time [15]. Operations of the system are modelled as a sequence of events in time, where each event marks a state change. When an *event* occurs, the system changes state according to the event and its current state. Classic examples of events include the arrival or sending of a message, function call or completion, etc. Discrete event simulation processes the events and their impact on system state entirely sequentially. Compared to other approaches (like register-transfer simulation), discrete event simulation operates on a higher level of abstraction.

In order for PDES to execute a single simulation environment in parallel, the model of the system needs to be split into separate instances called *Logical Processes* (LPs). Figure 4 illustrates the concept of LPs in a network of computer systems. The system consists of multiple clusters of computer systems connected with each other through a communication network. Each cluster of systems becomes a separate LP, which then is executed separately from the others.

When the LPs execute separately from each other, *synchronisation* problems might occur when they need to interact [15]. The processes might be at different simulated timestamps, which could result in an event arriving from another LP which occurred in the past relative to the current simulation time of the receiving process. A synchronisation mechanism is therefore required in order

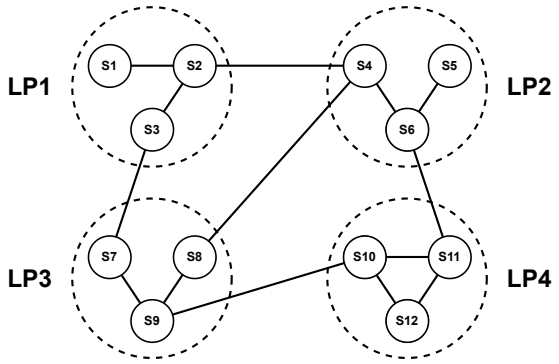

Figure 4: Illustration of the Logical Processes (LPs) concept in Parallel Discrete Event Simulation (PDES) applied to a network of systems.

to keep the PDES simulation consistent. The following subsections will discuss some of the research and state-of-the-art on the different synchronisation methodologies available for PDES.

3.2.1 Conservative Synchronisation. The first PDES synchronisation methodology is the concept of conservative synchronisation [15] [17] [41]. In conservative synchronisation, an LP will only continue its simulation if it can guarantee that no external events arrive between its current timestamp and the new future timestamp. The system has to guarantee that all events happen in order and that no LP can receive an event that occurred in the past of that LP its simulated time. If an LP would process an event before an earlier event has been processed, inconsistencies could occur.

Fujimoto [14] identifies two generations of conservative synchronisation algorithms. The first generation of algorithms has been introduced by Chandy, Misra, and Bryant [5] [9] in the late 1970s. Their algorithm blocks an LP's execution until it can guarantee that an event with a smaller timestamp can no longer be received. The system makes use of a communication framework with LP-to-LP links and an ordered (based on timestamp) message delivery guarantee. Using these LP-to-LP links, an LP can inspect those links to determine whether it can proceed with its execution. Whenever one of the links is empty, the LP blocks, as it can no longer guarantee that a message with a smaller timestamp is received. However, a deadlock might occur where each LP is waiting for another LP in a cycle. The authors propose the usage of *null* messages as a solution to the deadlock problem. A null message from an LP to another LP provides a guarantee that indicates the smallest timestamp value of any message it will later send on that link. This solution prevents the deadlock problem. However, it creates the lookahead creep problem [12].

The second-generation of conservative synchronisation algorithms addresses the lookahead creep problem with the key insight that the LPs have to know the timestamp of the next unprocessed event in the system. When the LPs know the timestamp of the next unprocessed event in the system, they can proceed to that timestamp without breaking the consistency of the simulation. Many other approaches to prevent deadlocks and lookahead creep problems were developed. Those solutions include, among others, an algorithm that detects deadlocks and resolves them [8], barrier synchronisation [14], and the usage of time intervals [13].

Since its first introduction in 1970, some research has been conducted on the implementation of conservative synchronisation and many tools have adopted this concept of PDES. A popular simulation environment which makes use of this approach is, again, the Omnet++ simulator [52]. The PDES implementation of Omnet++ actually makes use of the algorithm proposed by Chandy, Misra, and Bryant discussed earlier [51]. Another well-known discrete event simulation library called SystemC also provides conservative PDES functionality, called Time-Decoupled SystemC [55]. The GEM5 simulator has also implemented PDES using a conservative synchronisation methodology [35]. In GEM5, the approach differs slightly from the approach utilised by Omnet++. GEM5 uses a second-generation conservative approach in the form of a barrier for quantum-based (time interval) synchronisation. The simulator is supported by and utilised by many companies, such as the National Science Foundation, AMD, ARM, Hewlett-Packard, IBM, Intel, Metempsy, Micron, MIPS, Samsung, and Sun [19] [50].

3.2.2 Optimistic Synchronisation. The second PDES synchronisation methodology is the concept of optimistic synchronisation. In contrast to conservative synchronisation, where violations of the consistency constraint are avoided, the optimistic synchronisation approach allows those violations to occur and is able to detect and recover from them [15] [17] [41].

The optimistic synchronisation was introduced in 1985 by Jefferson and Sowizral [28]. Their approach - called "Time Warp" algorithm - allows errors (incorrect ordering of events) to occur, but it uses a rollback mechanism to reconstruct the execution of events in the correct order. Many advanced algorithms based on the Time Warp philosophy have since been proposed; e.g. Global Virtual Time (GVT) [18], direct cancellation [16], or lazy cancellation [18]. They are collectively known as optimistic synchronisation [12].

The Time Warp algorithm is based on two mechanisms, the local and the global control mechanism. The local control mechanism is responsible for the rollback within the LPs. The state of the LP can be recovered in the following three ways; (a) copy state saving, (b) incremental state saving, or (c) reverse computation. Strategy (a) can be expensive in both time and memory, as it copies the state to memory after every event. An alternative is the incremental state saving (b), where the copy of a state variable is only made before the first time an event modifies it. The last approach is the inverse computation (c), which performs a rollback using the inverse of the events. However, the inverse computation is not always possible and it therefore has to fallback on the incremental state saving [12]. A rollback can also influence the messages an LP has sent. The Time Warp algorithm makes use of *anti*-messages to undo a sent message. An anti-message is the exact same as the original message, but it has a flag enabled to signal it is an anti-message. Upon receiving, the LP can perform a rollback if the original message was already processed.

The global control mechanism addresses the problem of recovering memory utilised to hold the checkpointed states, and any event that has been processed in case it has to be reprocessed later. Several algorithms for this mechanism were proposed and developed [12].

Omnet++ was already presented as an example of a simulator which facilitates the conservative synchronisation approach. Nevertheless, it also supports using optimistic synchronisation. However, this requires writing significantly more complex code and the implementation of a more complicated simulation kernel [51]. Even when this is all in place, optimistic synchronisation may be slow when excessive rollbacks frequently occur.

3.2.3 Adaptive Synchronisation. The third PDES synchronisation methodology identified is the concept of adaptive synchronisation, which combines the previously discussed conservative and optimistic approaches [41]. The system can adapt its synchronisation technique based on the state of the system. It might be advantageous to utilise optimistic synchronisation when the system tolerates it, or switch to conservative when the rollback is difficult (or infeasible in the case of irreversible computations). Various concepts to adaptive synchronisation have been proposed [29] [40] [44].

3.3 Hardware Accelerators

The third methodology is the concept of *Hardware Acceleration*. In the last decades, increasingly more research has been conducted on the usage of hardware acceleration to speed up and scale applications. Well-known examples of hardware accelerators are *Graphics Processing Units* (GPUs), *Application-Specific Integrated Circuits* (ASICs), and *Field-Programmable Gate Arrays* (FPGAs).

Typically, software applications run on a general-purpose central processing unit (CPU). As the CPU is designed to be general-purpose, it has to provide for a wide range of functionalities. Hardware accelerators are not restricted by this constraint and can hence aim to perform specific functions more efficiently.

A well-known hardware accelerator is the Graphics Processing Unit (GPU). Originally, GPUs were designed to accelerate the rendering of 3D graphics. Over time, the platform grew more flexible and programmable, becoming more capable in other fields, such as Parallel Computing, Deep Learning, and High-Performance Computing (HPC) [27].

At a high-level, the GPU can be described as a large collection of (simpler) compute resources when compared to a CPU, which has fewer resources but they are more complex. GPUs are classified as manycore processors, whereas current generation (consumer grade) CPUs typically are classified as multicore processors. There is a different philosophy between the two classifications, where manycore optimises explicit parallelism and throughput by trading in the latency and single thread performance from multicore. As the GPU has many compute resources, it can perform many simple tasks concurrently, yielding a high throughput.

A different kind of hardware accelerator, the Application-Specific Integrated Circuit (ASIC), is an integrated circuit chip specially designed for a particular functionality or application. ASIC chips are typically smaller, more power efficient and outperform general-purpose processors. However, the non-recurring engineering (NRE) cost of designing an ASIC is significant. A large production volume is often required to justify and amortise the costs [31].

Another established hardware accelerator is the FPGA (Field-Programmable Gate Array). An FPGA is an integrated circuit configurable by a user after it has been manufactured. Typically, a Hardware Description Language (HDL) is used to configure the integrated circuit. Compared to ASICs, FPGAs are not application-specific and can be utilised in many different applications. The most notable feature of the FPGA is in its name, "Field-Programmable". This means that it can be reconfigured and adapted as is desired, without having the NRE costs of engineering that would still be required for a fixed circuit chip. Typically, FPGAs are used for prototyping and low-quantity production volumes, whereas ASICs are used for large production volumes [31].

Another advantage of the reconfigurability of the FPGAs is automation. The FPGA can adapt its configuration to suit the current needs of an application through software, without the need to replace it for a different component or manual setup. One area where FPGAs have been used, is in the design exploration of SoCs and microprocessors. Usually, the models of those systems fit on the FPGA and allow the designer to adapt, test, and validate their designs in a timely and automated manner [10]. However, simulating an entire dCPS on an FPGA is likely not feasible, as those systems are simply too large to fit (assuming the abstraction level of the dCPS model is not very high) [2]. However, multiple FPGAs could co-operate, where each FPGA can be used to simulate submodules of the dCPS model. Instead of imitating a part of the dCPS to accelerate the simulation, the hardware could also be used to accelerate (parts of) the functionalities from the simulator itself.

Already in the 80s, Blank conducted a survey on hardware accelerators in the computer-aided design domain (CAD tools) [3]. A set of machines using hardware accelerators at that time is identified. For each machine, the architecture and the relative performance are discussed. It is concluded that the choice to integrate hardware accelerators depends on many factors like cost, software support, and maintenance.

In recent decades, the increasing usability, utility and availability of hardware accelerators are making them more attractive and applicable in a broad range of research domains. Xiao et al. [56] discuss the value hardware accelerators can bring to agent-based simulations (applicable to fields such as road traffic, social networks, military, biology, and economics). The survey categorised existing approaches based on the key challenges of hardware assignment (i.e., scattered memory accesses, abstraction from hardware specifics, and more). Their main observations are that (i) most of the literature in the past years has focused on GPUs and it is expected that a significant amount of work on FPGAs for agent-based simulations will appear in the near future. (ii) A vast amount of work has proposed techniques for efficient execution of agent-based simulations, but only few techniques have been incorporated into a unified framework. Finally, they sketch a vision of a framework that includes automated hardware mapping and performance optimisation.

Rahman, Abu-Ghazaleh, and Najjar [43] propose a general accelerator framework for PDES implemented on FPGAs that can specialise to any particular simulation model. The architecture allows multiple accelerators to be connected with each other to scale up the simulation. The article explores several design alternatives and expresses the trade-offs between the different options. Their future work aims to address three different directions. First, they intent to reduce the impact of memory access time and resource contention on the architecture. Second, they want to research multiple accelerators working together on larger models. Finally, the

development of a programming environment to provide rapid prototyping.

4 DISCUSSION

The related works presented the state-of-the-art on methods and techniques for the scalability of system-level simulation environments. In this section, the methodologies will be reviewed in the context in which the literature study is conducted. Additionally, the techniques will be discussed in each other's context and their characteristics will be examined.

4.1 Simulation Campaigns

The first approach to scalability discussed in the related works is the concept of simulation campaigns. To provide a brief synopsis, simulation campaigns have the capability to evaluate multiple design points at the same time. Typically, this is achieved by distributing multiple (independent) simulation environments on separate resources.

A way to characterise scalability, as defined in Section 2, is to express it by capabilities (size, time, and complexity). In the case of simulation campaigns, the size capability best aligns, as the approach aims to increase the number of concurrent simulations. Time capability is not addressed, since the individual simulations do not achieve speedup through simulation campaigns. The complexity capability is not clearly part of its philosophy. For example, it could improve precision in a stochastic simulation by performing more iterations in the same time interval (related to size capability). However, the individual simulations themselves are not more extensive or complex.

Applying a scalability concept to a system is not straightforward. When considering simulation campaigns in the context of DSE2.0, all its characteristics, intricacies, and capabilities must be considered. The design space exploration algorithm also impacts the compatibility of a scalability approach. Simulation campaigns benefit from concurrent design point evaluation, focusing on improved throughput. Therefore, an iterative batching exploration stage in stage (iii) of the general DSE2.0 workflow, see Figure 1, would be a good candidate to apply simulation campaigns. However, a designer of a dCPS will not be provided with improved latency on single evaluations, which could be utilised to guide the DSE manually. It also means that DSE algorithms processing design points one by one will not be able to take advantage of simulation campaigns. Additionally, a hardware dependency is present in the solution. Theoretically, the solution could scale infinitely, only bounded by the number of resources available. Nonetheless, when the model of the dCPS becomes too large to be processed on a single resource, simulation campaigns might not be able to provide its scalability property.

After the approach is deemed applicable in the context of the system, the scalability service needs to be implemented and maintained. From a purely theoretical point of view, it is deemed that the simulation campaigns is the most straightforward scalability approach, both as a concept and to implement. When a single simulation environment can successfully be deployed, the simplest form of campaigns only replicates the same process on different resources and collects the results afterwards. One benefit of this approach is that many existing simulation environments and tools, like Omnet++ and Akaroa, already have the simulation campaign functionality built-in.

Looking back at the second scientific challenge defined by Herget et al. [22], scalability was desired to address the significant expansion of the design space caused by the increased number of design choices that come with the complexity and heterogeneity of dCPS. They envisioned a scalable and efficient approach that utilises the scalability of the simulation environment or the scalability of search and pruning strategies. Simulation campaigns offer scalability from within the simulation environment, adapting to the number of design points generated by the search strategy.

4.2 Parallel Discrete Event Simulation (PDES)

The second approach is Parallel Discrete Event Simulation (PDES). In this methodology, the model of a design point is split into Logical Processes (LPs), which will be executed on separate resources. As the processes will run independently, a synchronisation mechanism is required to keep consistency with the sequential simulation. Multiple techniques were identified: conservative synchronisation, optimistic synchronisation, and adaptive synchronisation.

In the case of PDES, the time and complexity capabilities of the scalability definition in Section 2 are applicable. It is able to increase the performance of a single simulation by distributing it over multiple resources. Additionally, it can handle more complex and extensive models. Whereas the simulation campaign approach has size capability, PDES does not, as it only applies to a single design point.

The difference in scalability capabilities will be visible in the application of PDES in the context of DSE2.0. Instead of throughput, the main feature of PDES is latency and model complexity. PDES does not adapt to the number of design points generated by the search strategy. However, when a designer needs a quick answer to a "What-if" question, PDES can provide a (relatively) low latency response. Additionally, PDES can provide support for large dCPS models, which could not be processed on a single resource due to hardware limitations (i.e., memory requirements). A DSE algorithm which processes design points one by one would benefit most from this approach, as it offers no size capability for batches.

Implementing PDES, from a theoretical standpoint, is more complicated than simulation campaigns. The system needs to take into account that changing design points affect the model and LPs. Ideally, the LPs are automatically inferred from a design point, or manually designated based on a high-level system model that can transfer the LP partitioning to the design points. Additionally, a synchronisation mechanism needs to be in place to guarantee consistency. As was the case with simulation campaigns, a wide range of simulation environments and tools, like Omnet++ and GEM5, have PDES built-in.

Parallel Discrete Event Simulation addresses the scalability challenge from Herget et al. [22] solely for the simulation environment by distributing a single model over multiple resources.

4.3 Hardware Accelerators

Lastly, hardware accelerators were discussed as a potential scalability approach. Three different hardware accelerators were considered; (i) Graphics Processing Units (GPUs), (ii) Application-Specific Integrated Circuits (ASICs), and (iii) Field-Programmable Gate Arrays (FPGAs). The philosophy behind these techniques is that, instead of using a general purpose processor, specialised hardware aiming to perform specific functions more efficiently is used.

It is not straightforward to classify the scalability capabilities of hardware accelerators, as it depends on the integration into the system. Hardware accelerators change the platform on which simulations can be performed, thereby having the potential to achieve any of three capabilities (size, time, and complexity) or combinations thereof. The large number of resources provided by a GPU could be used to execute more simulations simultaneously, achieving size capability. ASICs and FPGAs could provide time and complexity capabilities by specialising for a specific (complex) model, improving performance and facilitating more complex simulations.

As the capabilities of hardware accelerators are not clearly defined, it also results in more difficulties to place them in the context of DSE2.0. In principle, hardware accelerators significantly change the platform on which DSE will be performed. This makes it the least straightforward scalability approach of the three methodologies, both as a concept and to implement. The platform allows an entirely custom deployment of the DSE workflow on specialised hardware. Hardware accelerators could be used in several ways in the DSE workflow; offloading computation on specialised hardware, concurrent simulations, and more.

When it is deemed the right fit in the DSE process, the system needs to be adapted to utilise the hardware accelerator(s) properly. As the platform provides a broad range of customisability, integrating it into an established environment is a complex and time-consuming process. The hardware accelerators approach is therefore believed to be the most complicated from the three methods discussed. Research is being performed on utilising hardware accelerators in simulation environments [56] [43]. However, the approach is not readily available in existing popular simulators.

Hardware accelerators address the scalability challenge from Herget et al. [22] in the simulation environment, but may also affect the implementation of the search and pruning strategies in the DSE.

4.4 Composite Solutions

The previously discussed techniques all approach the problem of scalability differently, are applicable to specific use cases, and have a set of requirements. However, scalability of system-level simulation environments is not necessarily solved by exclusively using one single methodology. Research has also been conducted on, or made use of, combinations of the different techniques. Following is a brief highlight of several composite solutions, which includes: Simulation Campaigns with PDES, Cloning, Cloning with hardware accelerators, and PDES with hardware accelerators.

Simulation campaigns do not have to operate solely with sequential simulations, the technique can also be used with PDES. By uniting campaigns and PDES in the DSE process, the designer is provided with improved throughput and latency. This solution could also alleviate the hardware limitations that could occur with simulation campaigns when, for example, the models are too large for the memory available on a single resource.

Another related technique is called "Cloning" [24] [4] [26], where the designer can interject a *decision point* into an ongoing simulation. At this decision point, the system executes a simulation campaign on a tree of "what-if" scenarios. When applying this to DSE, a good fit would be a search strategy that sequentially evaluates design points. For example, when the designer interjects a decision point into an ongoing PDES simulation, a campaign of PDES simulations is deployed. This could provide the designer with more fine-grained control on the direction of the DSE. On top of cloning, Yoginath and Perumalla [57] have proposed a combination with hardware accelerators. By applying cloning on large-scale GPU platforms, the DSE process might benefit from even more scalability.

Another combination is the application of PDES on hardware accelerators through a general framework [43], combining the improved latency of PDES and the specialisation of hardware accelerators. A general framework, such as the one proposed in [43], to integrate hardware accelerators could provide a platform for research, like DSE2.0, to better assess the capabilities of hardware accelerators and whether they are applicable to their solutions.

5 CONCLUSION

This literature study discussed the state-of-the-art on methods and techniques for the scalability of system-level simulation environments. These scalability techniques were considered in the context of the DSE2.0 project, where the entirety of the state-of-the-art in Design Space Exploration is under scrutiny. DSE2.0 identified two challenges in advancing the field towards efficient and scalable DSE for distributed Cyber-Physical Systems; (i) Modelling complex dCPS, and (ii) Scalable DSE.

Especially challenge (ii) was of interest to this literature study. Addressing the scalability challenge started with a definition. Scalability was expressed in size, time and complexity capabilities. The related works identified and analysed three approaches to scalability in simulation environments; (i) Simulation Campaigns, (ii) Parallel Discrete Event Simulations, and (iii) Hardware Accelerators. The philosophy and capabilities of the three approaches significantly differ, making them applicable to different system architectures. Additionally, composite solutions were briefly discussed as they can provide a middle ground between approaches and provide improved services.

Based on the related works and discussion on the three individual techniques, simulation campaigns would be the first suggestion for scalability in the context of DSE2.0. It is a simple concept which provides increased throughput, adapts to the number of tasks and resources, and many state-of-the-art simulators have adopted the technique. However, a combination of simulation campaigns and PDES could be a good candidate, when composite solutions are also considered. The composition can provide improved throughput and latency, whilst also supporting more extensive system models. Hardware accelerators are not included in the suggested approach. Although it could improve the scalability, it also adds significant complexity to the infrastructure and implementation. Therefore, it

is not suggested as the initial approach to scalability in the context of DSE2.0.

Nevertheless, further research still has to be conducted to define, implement, and analyse the concepts and specific workflows that contribute to the scalability of system-level simulation environments in the context of the DSE2.0 project. Assessing the impact on the entirety of the DSE workflow can provide insights into possible optimisations to further the facilitation of design space exploration for dCPS.

REFERENCES

[1] Benny Akesson, Mitra Nasri, Geoffrey Nelissen, Sebastian Altmeyer, and Robert I. Davis. "A comprehensive survey of industry practice in real-time systems". In: *Real-Time Systems* (2021). Publisher: Springer, pp. 1–41.

[2] Sameh Asaad, Ralph Bellofatto, Bernard Brezzo, Chuck Haymes, Mohit Kapur, Benjamin Parker, Thomas Roewer, Proshanta Saha, Todd Takken, and José Tierno. "A cycle-accurate, cycle-reproducible multi-FPGA system for accelerating multi-core processor simulation". In: *Proceedings of the ACM/SIGDA international symposium on Field Programmable Gate Arrays*. 2012, pp. 153–162. DOI: 10.1145/2145694.2145720.

[3] Tom Blank. "A survey of hardware accelerators used in computer-aided design". In: *IEEE Design & Test of Computers* 1.3 (1984). Publisher: IEEE, pp. 21–39. DOI: 10.1109/MDT.1984.5005647.

[4] Luciano Bononi, Michele Bracuto, Gabriele D'Angelo, and Lorenzo Donatiello. "Concurrent replication of parallel and distributed simulations". In: *Workshop on Principles of Advanced and Distributed Simulation (PADS'05)*. IEEE, 2005, pp. 234–243. DOI: 10.1109/PADS.2005.6.

[5] Randal Everitt Bryant. *Simulation of Packet Communication Architecture Computer Systems*. MASSACHUSETTS INST OF TECH CAMBRIDGE LAB FOR COMPUTER SCIENCE, 1977.

[6] Anastasiia Butko, Rafael Garibotti, Luciano Ost, and Gilles Sassatelli. "Accuracy evaluation of gem5 simulator system". In: *7th International workshop on reconfigurable and communication-centric systems-on-chip (ReCoSoC)*. IEEE, 2012, pp. 1–7. DOI: 10.1109/ReCoSoC.2012.6322869.

[7] Lukai Cai and Daniel Gajski. "Transaction level modeling: an overview". In: *First IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and Systems Synthesis (IEEE Cat. No. 03TH8721)*. IEEE, 2003, pp. 19–24.

[8] K. Mani Chandy and Jayadev Misra. "Asynchronous distributed simulation via a sequence of parallel computations". In: *Communications of the ACM* 24.4 (1981). Publisher: ACM New York, NY, USA, pp. 198–206. DOI: 10.1145/358598.358613.

[9] K. Mani Chandy and Jayadev Misra. "Distributed simulation: A case study in design and verification of distributed programs". In: *IEEE Transactions on software engineering* 5 (1979). Publisher: IEEE, pp. 440–452. DOI: 10.1109/TSE.1979.230182.

[10] André DeHon and John Wawrzynek. "Reconfigurable computing: what, why, and implications for design automation". In: *Proceedings of the 36th annual ACM/IEEE Design Automation Conference*. 1999, pp. 610–615.

[11] Greg Ewing, Krzysztof Pawlikowski, and Don McNickle. "Akaroa-2: Exploiting network computing by distributing stochastic simulation". In: (1999). Publisher: SCSI Press.

[12] Richard Fujimoto. "PARALLEL AND DISTRIBUTED SIMULATION". In: *Proceedings of the 2015 Winter Simulation Conference* (2015), p. 15.

[13] Richard M. Fujimoto. *Parallel and distributed simulation systems*. Vol. 300. New York: Wiley, 2000.

[14] Richard M. Fujimoto. "Parallel and distributed simulation systems". In: *Proceeding of the 2001 Winter Simulation Conference (Cat. No. 01CH37304)*. Vol. 1. IEEE, 2001, pp. 147–157. DOI: 10.1109/WSC.2001.977259.

[15] Richard M. Fujimoto. "Parallel discrete event simulation". In: *Communications of the ACM* 33.10 (1990). Publisher: ACM New York, NY, USA, pp. 30–53. DOI: 10.1145/84537.84545.

[16] Richard M. Fujimoto. *Time Warp on a shared memory multiprocessor*. UTAH UNIV SALT LAKE CITY SCHOOL OF COMPUTING, 1989.

[17] Richard M. Fujimoto, Rajive Bagrodia, Randal E. Bryant, K. Mani Chandy, David Jefferson, Jayadev Misra, David Nicol, and Brian Unger. "Parallel discrete event simulation: The making of a field". In: *2017 Winter Simulation Conference (WSC)*. IEEE, 2017, pp. 262–291. DOI: 10.1109/WSC.2017.8247793.

[18] Richard M. Fujimoto and Maria Hybinette. "Computing global virtual time in shared-memory multiprocessors". In: *ACM Transactions on Modeling and Computer Simulation (TOMACS)* 7.4 (1997). Publisher: ACM New York, NY, USA, pp. 425–446. DOI: 10.1145/268403.268404.

[19] GEM5. *gem5: About*. URL: https://www.gem5.org/about/ (visited on 11/23/2023).

[20] Andreas Hansson, Benny Akesson, and Jef Van Meerbergen. "Multi-processor programming in the embedded system curriculum". In: *ACM SIGBED Review* 6.1 (2009). Publisher: ACM New York, NY, USA, pp. 1–9.

[21] Philip Heidelberger. "Statistical analysis of parallel simulations". In: *Proceedings of the 18th conference on Winter simulation*. 1986, pp. 290–295. DOI: 10.1145/318242.318448.

[22] Marius Herget, Faezeh Sadat Saadatmand, Martin Bor, Ignacio González Alonso, Todor Stefanov, Benny Akesson, and Andy D. Pimentel. "Design Space Exploration for Distributed Cyber-Physical Systems: State-of-the-art, Challenges, and Directions". In: 2022 25th Euromicro Conference on Digital System Design (DSD). IEEE, 2022, pp. 632–640. ISBN: 978-1-66547-404-7. DOI: 10.1109/DSD57027.2022.00090.

[23] Mark D. Hill. "What is scalability?" In: *ACM SIGARCH Computer Architecture News* 18.4 (1990). Publisher: ACM New York, NY, USA, pp. 18–21. DOI: 10.1145/121973.121975.

[24] Maria Hybinette and Richard Fujimoto. "Cloning: A novel method for interactive parallel simulation". In: *Proceedings of the 29th conference on Winter simulation*. 1997, pp. 444–451. DOI: 10.1145/268437.268523.

[25] Maria Hybinette and Richard M. Fujimoto. "Cloning parallel simulations". In: *ACM Transactions on Modeling and Computer Simulation (TOMACS)* 11.4 (2001). Publisher: ACM New York, NY, USA, pp. 378–407. DOI: 10.1145/508366.508370.

[26] Maria Hybinette and Richard M. Fujimoto. "Scalability of parallel simulation cloning". In: *Proceedings 35th Annual Simulation Symposium. SS 2002*. IEEE, 2002, pp. 275–282. DOI: 10.1109/SIMSYM.2002.1000164.

[27] Intel. *What Is a GPU? Graphics Processing Units Defined*. Intel. Dec. 23, 2022. URL: https://www.intel.com/content/www/us/en/products/docs/processors/what-is-a-gpu.html (visited on 12/27/2022).

[28] David R. Jefferson. "Virtual time". In: *ACM Transactions on Programming Languages and Systems (TOPLAS)* 7.3 (1985). Publisher: ACM New York, NY, USA, pp. 404–425. DOI: 10.1145/3916.3988.

[29] Vikas Jha and Rajive L. Bagrodia. "A unified framework for conservative and optimistic distributed simulation". In: *ACM SIGSIM Simulation Digest* 24.1 (1994). Publisher: ACM New York, NY, USA, pp. 12–19. DOI: 10.1145/195291.182480.

[30] Minyoung Kim, Sudarshan Banerjee, Nikil Dutt, and Nalini Venkatasubramanian. "Design space exploration of real-time multi-media MPSoCs with heterogeneous scheduling policies". In: *Proceedings of the 4th international conference on Hardware/software codesign and system synthesis*. 2006, pp. 16–21. DOI: 10.1145/1176254.1176261.

[31] Jeff Kriegbaum. *FPGA's vs. ASIC's - EE Times*. Sept. 13, 2004. URL: https://www.eetimes.com/fpgas-vs-asics/ (visited on 12/27/2022).

[32] Karthik Lakshmanan, Shinpei Kato, and Ragunathan Rajkumar. "Scheduling parallel real-time tasks on multi-core processors". In: *2010 31st IEEE Real-Time Systems Symposium*. IEEE, 2010, pp. 259–268. DOI: 10.1109/RTSS.2010.42.

[33] Darren R. Law. "Scalable means more than more: a unifying definition of simulation scalability". In: *1998 Winter Simulation Conference. Proceedings (Cat. No. 98CH36274)*. Vol. 1. IEEE, 1998, pp. 781–788. DOI: 10.1109/WSC.1998.745064.

[34] Doug Lea. "A java fork/join framework". In: *Proceedings of the ACM 2000 conference on Java Grande*. 2000, pp. 36–43. DOI: 10.1145/337449.337465.

[35] Jason Lowe-Power, Abdul Mutaal Ahmad, Ayaz Akram, Mohammad Alian, Rico Amslinger, Matteo Andreozzi, Adrià Armejach, Nils Asmussen, Brad Beckmann, and Srikant Bharadwaj. "The gem5 simulator: Version 20.0+". In: *arXiv preprint arXiv:2007.03152* (2020). DOI: 10.48550/arxiv.2007.03152.

[36] Edward A. Luke. "Defining and measuring scalability". In: *Proceedings of Scalable Parallel Libraries Conference*. IEEE, 1993, pp. 183–186.

[37] Brit Meier, Mladen Skelin, Frans Beenker, and Wouter Leibbrandt. *HTSM Systems Engineering Roadmap*. July 24, 2020.

[38] Edjair Mota, Adam Wolisz, and Krzysztof Pawlikowski. "A perspective of batching methods in a simulation environment of multiple replications in parallel". In: *2000 Winter Simulation Conference Proceedings (Cat. No. 00CH37165)*. Vol. 1. IEEE, 2000, pp. 761–766.

[39] Krzysztof Pawlikowski, Victor WC Yau, and Don McNickle. "Distributed stochastic discrete-event simulation in parallel time streams". In: *Proceedings of Winter Simulation Conference*. IEEE, 1994, pp. 723–730. DOI: 10.1109/WSC.1994.717420.

[40] Kalyan S. Perumalla. "/spl mu/sik-a micro-kernel for parallel/distributed simulation systems". In: *Workshop on Principles of Advanced and Distributed Simulation (PADS'05)*. IEEE, 2005, pp. 59–68. DOI: 10.1109/PADS.2005.1.

[41] Kalyan S. Perumalla. "Parallel and distributed simulation: traditional techniques and recent advances". In: *2006 Winter Simulation Conference*. IEEE Computer Society, 2006, pp. 84–95. DOI: 10.1109/WSC.2006.323041.

[42] Andy D. Pimentel. "Exploring exploration: A tutorial introduction to embedded systems design space exploration". In: *IEEE Design & Test* 34.1 (2016). Publisher: IEEE, pp. 77–90. DOI: 10.1109/MDAT.2016.2626445.

[43] Shafiur Rahman, Nael Abu-Ghazaleh, and Walid Najjar. "PDES-A: Accelerators for parallel discrete event simulation implemented on FPGAs". In: *ACM Transactions on Modeling and Computer Simulation (TOMACS)* 29.2 (2019). Publisher: ACM New York, NY, USA, pp. 1–25. DOI: 10.1145/3302259.

[44] Hassan Rajaei, Rassul Ayani, and Lars-Erik Thorelli. "The local Time Warp approach to parallel simulation". In: *Proceedings of the seventh workshop on*

Parallel and distributed simulation. 1993, pp. 119–126. DOI: 10.1145/158459.158474.

[45] Paul Rosenfeld, Elliott Cooper-Balis, and Bruce Jacob. "DRAMSim2: A cycle accurate memory system simulator". In: *IEEE computer architecture letters* 10.1 (2011). Publisher: IEEE, pp. 16–19. DOI: 10.1109/L-CA.2011.4.

[46] Sartaj Sahni and George Vairaktarakis. "The master-slave paradigm in parallel computer and industrial settings". In: *Journal of Global Optimization* 9.3 (1996). Publisher: Springer, pp. 357–377. DOI: 10.1007/BF00121679.

[47] Bram van der Sanden, Yonghui Li, Joris van den Aker, Benny Akesson, Tjerk Bijlsma, Martijn Hendriks, Kostas Triantafyllidis, Jacques Verriet, Jeroen Voeten, and Twan Basten. "Model-Driven System-Performance Engineering for Cyber-Physical Systems: Industry Session Paper". In: *2021 International Conference on Embedded Software (EMSOFT)*. IEEE, 2021, pp. 11–22. DOI: 10.1145/3477244.3477985.

[48] G. S. Sangeetha, Vignesh Radhakrishnan, Prabhu Prasad, Khyamling Parane, and Basavaraj Talawar. "Trace-driven simulation and design space exploration of network-on-chip topologies on FPGA". In: *2018 8th International Symposium on Embedded Computing and System Design (ISED)*. IEEE, 2018, pp. 129–134. DOI: 10.1109/ISED.2018.8703884.

[49] Simon Streltsov and Pirooz Vakili. "Parallel replicated simulation of markov chains: implementation and variance reduction". In: *Proceedings of the 25th conference on Winter simulation*. 1993, pp. 430–436. DOI: 10.1145/256563.256682.

[50] UC Davis. *Simulation Research and gem5*. UC Davis Computer Architecture. URL: https://arch.cs.ucdavis.edu/projects/gem5 (visited on 11/23/2022).

[51] Andras Varga. *OMNeT++ discrete event simulation system version 6.x user manual*. 2022. URL: https://doc.omnetpp.org/omnetpp/SimulationManual.pdf (visited on 11/13/2022).

[52] András Varga and Rudolf Hornig. "An overview of the OMNeT++ simulation environment". In: *1st International ICST Conference on Simulation Tools and Techniques for Communications, Networks and Systems*. 2010. DOI: 10.4108/icst.simutools2008.3027.

[53] Guangxi Wan and Peng Zeng. "Codesign of Architecture, Control, and Scheduling of Modular Cyber-Physical Production Systems for Design Space Exploration". In: *IEEE Transactions on Industrial Informatics* 18.4 (2021). Publisher: IEEE, pp. 2287–2296. DOI: 10.1109/TII.2021.3097761.

[54] Charles B. Weinstock and John B. Goodenough. *On system scalability*. Software Engineering Institute, Carnegie Mellon University, 2006.

[55] Jan Henrik Weinstock, Christoph Schumacher, Rainer Leupers, Gerd Ascheid, and Laura Tosoratto. "Time-decoupled parallel SystemC simulation". In: *2014 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2014, pp. 1–4.

[56] Jiajian Xiao, Philipp Andelfinger, David Eckhoff, Wentong Cai, and Alois Knoll. "A survey on agent-based simulation using hardware accelerators". In: *ACM Computing Surveys (CSUR)* 51.6 (2019). Publisher: ACM New York, NY, USA, pp. 1–35. DOI: 10.1145/3291048.

[57] Srikanth B. Yoginath and Kalyan S. Perumalla. "Scalable cloning on large-scale gpu platforms with application to time-stepped simulations on grids". In: *ACM Transactions on Modeling and Computer Simulation (TOMACS)* 28.1 (2018). Publisher: ACM New York, NY, USA, pp. 11–26. DOI: 10.1145/3158669.

This literature study has been produced as part of the XM_0131 course at the Vrije Universiteit Amsterdam in cooperation with the University of Amsterdam.

Supervisors:

Prof. Dr. Andy D. Pimentel
University of Amsterdam
a.d.pimentel@uva.nl

Prof. Dr. Benny Akesson
University of Amsterdam & TNO-ESI
k.b.akesson@uva.nl

Marius Herget
University of Amsterdam
m.herget@uva.nl