



Universiteit
Leiden
The Netherlands

Towards design space exploration of industrial distributed cyber-physical systems

Saadatmand, F.S.

Citation

Saadatmand, F. S. (2026, September 30). *Towards design space exploration of industrial distributed cyber-physical systems*. Retrieved from <https://hdl.handle.net/1887/4310251>

Version: Publisher's Version

License: [Licence agreement concerning inclusion of doctoral thesis in the Institutional Repository of the University of Leiden](#)

Downloaded from: <https://hdl.handle.net/1887/4310251>

Note: To cite this publication please use the final published version (if applicable).

Chapter 5

CompDSE Evaluation

Publications related to this chapter

- Faezeh Sadat Saadatmand, Todor Stefanov, Ignacio González Alonso, Andy D. Pimentel, and Benny Akesson. CompDSE: A Methodology for Design Space Exploration of Computing Subsystems Within Complex Cyber-Physical Systems. *IET Cyber-Physical Systems: Theory & Applications*, 10(1), 2025.

In this chapter, we evaluate the CompDSE methodology using an industrial case study, namely the ASML TWINSKAN machine. The goal is to assess the efficiency and accuracy of the proposed methodology in supporting DSE of complex industrial dCPS, thereby addressing the efficiency aspect of challenge **CHL4**.

Section 5.1 highlights the need for evaluation, including the validation of both accuracy and efficiency. Section 5.2 outlines the contributions of this chapter, including the evaluation itself and a set of DSE experiments. Section 5.3 presents the experimental setup. This is followed by Section 5.4, which presents the experimental results, compares simulated timing with real-time measurements to assess accuracy, and records the simulation duration to assess the efficiency for evaluating the performance of a single design point. This section also explores alternative configurations of certain subsystems within the ASML TWINSKAN machine and compares their performance evaluation results. Finally, Section 5.5 concludes the chapter.

5.1 Problem Statement

The previous chapters introduced the CompDSE methodology, including the abstract models required for DSE of complex dCPS, the algorithms for their automated derivation from execution traces, and the generation of simulation-ready code. However, while the methodology, modeling approach, and supporting tools have been fully developed and described, their effectiveness has to be validated in a realistic industrial context. In particular, it is necessary to assess how accurately the derived models reflect the actual system behavior required for effective DSE, and how efficient the system simulation is for rapid performance evaluation of each design point.

5.2 Contributions

This chapter evaluates the CompDSE methodology using the ASML TWINSKAN lithography machine, introduced in Section 2.3, as a representative industrial dCPS case study. While the experiments are based on the TWINSKAN machine, the methodology and findings are applicable to other complex dCPS with similar architectural characteristics. The main contributions of this chapter are as follows:

1. **Accuracy validation:** We evaluate how closely the simulation results produced by the CompDSE models align with the actual TWINSKAN system behavior, thereby validating the fidelity of the modeling process. In particular, we compare the average time required for wafer processing in the simulated and real systems;
2. **Efficiency assessment:** We assess the time required to simulate a single design point, demonstrating the efficiency of the CompDSE methodology despite the complexity and scale of the TWINSKAN industrial dCPS;
3. **DSE experiments:** We conduct simulation-based DSE experiments to evaluate alternative architectural configurations of some TWINSKAN subsystems. These experiments aim to identify potential system performance improvements while respecting physical and operational constraints of the system.

5.3 Experimental Setup

To evaluate the accuracy and efficiency of the proposed methodology, we conducted an experiment based on real trace data collected from the ASML TWINSKAN machine

Table 5.1: Scale of the traced system and model derivation

Complexity Metric	Value
Number of traced hosts	8
Number of processes	445
Number of channels	2057

testbench. We traced a subset of the hosts in the testbench, including one Linux-based server and several real-time embedded systems running the Wind River VxWorks OS. Due to technical limitations such as some hosts being “bare-metal” (i.e., not running any operating system), it was not possible to trace all hosts. However, we consider the effects of these untraced hosts on the overall dCPS performance during our system simulations by using Δ parameters, as explained in Section 3.5.4.

To incorporate Δ parameters during simulation, the simulated timestamp of every *write* event, corresponding to a message sent to an untraced host, is recorded. When the simulator encounters a *read* event corresponding to a message received from an untraced host, it first computes the difference between the current simulation time and the recorded timestamp of the preceding *write* event. This calculated difference represents the elapsed time in the simulation since the *write* event occurred. The simulator then subtracts this elapsed time from the Δ parameter value corresponding to the *read* event in order to determine the remaining time that should be waited before scheduling the *read* event for execution. This approach ensures that the impact of the physical environment (including untraced hosts) on the dCPS timing performance is accurately represented in the simulation.

Table 5.1 summarizes the complexity metrics of the generated models, including the number of traced hosts, modeled processes, and identified communication channels. In this experiment, the CompDSE methodology is applied to a wafer lot operation involving five wafers. A total of **8** hosts are traced, **445** application processes are identified and modeled, and **2057** communication channels between these processes are identified. These channels are derived to support message exchange and synchronization, independent of how the hardware platform model is abstracted.

In our integrated simulation setup, synchronization is handled by the hardware platform layer. Messages exchanged between processes are transferred through the simulated hardware platform. When a process attempts to read a message, it can only proceed after receiving a confirmation from its allocated host, indicating that the message is now available for reading. As a result, the aforementioned communication

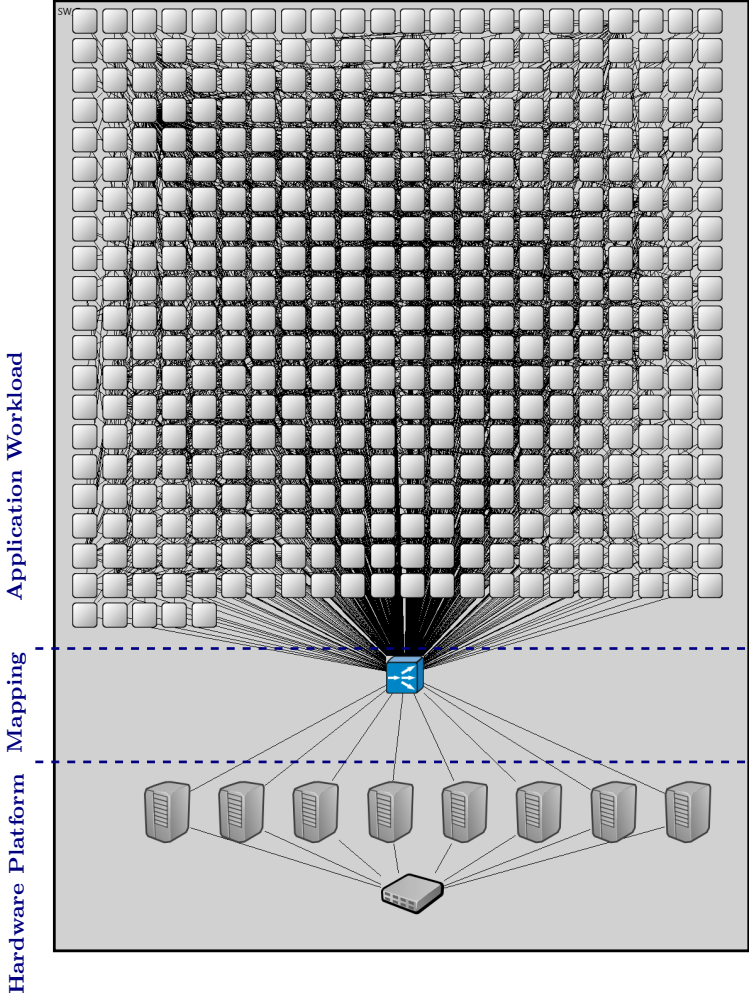


Figure 5.1: The simulation setup of the ASML TWINSCAN machine in OMNet++

channels serve as logical abstractions and are not explicitly represented in the NED file.

Figure 5.1 shows a visualization of the simulation environment used in our experiment for the TWINSCAN machine, implemented in OMNet++ and organized into three layers. The names of the entities (processes, hosts, etc.) are omitted due to confidentiality imposed by ASML. In the platform layer, the eight traced hosts used in our simulation are connected to a central switch.

5.4 Evaluation Results

After describing the experimental setup above, in this section we evaluate the model accuracy, simulation efficiency, and explore some alternative system configurations.

5.4.1 Efficiency

The efficiency of CompDSE is one of its core strengths, facilitating system designers to effectively explore the vast design space of computing subsystems within a complex dCPS. By considering the system complexity, shown in Table 5.1, the huge number of possibilities (in the range of 8^{445}) to map the 445 processes onto the 8 hosts creates a design space of extraordinary size. Furthermore, when considering additional factors - such as the number of cores per host, different operating frequencies, and various scheduling policies - the number of possible configurations expands into an astronomical number of combinations. These numbers highlight the high complexity of the derived model and the corresponding design space, which is typical for industrial-scale dCPS.

Table 5.2 provides the execution time of the various steps within CompDSE, visualized in Figure 3.1. The time required to manually add tracepoints in the software infrastructure depends on the number of tracepoints needed. For our case, we add tracepoints at the start and end of four functions within a shared library used by all processes. Adding these 8 tracepoints takes approximately 5 minutes and needs to be done only once. To ensure that tracing does not affect the system's real-time performance, we monitored the wafer processing time before and after enabling tracing and observed no measurable difference.

Afterward, the time required for trace collection depends on the duration of the application's execution. In our case, processing five wafers takes 4.33 minutes. This trace collection process must be repeated for each operation mode. After the trace collection, the automated model derivation and integration take 29.5 hours but needs to be performed only once. The model-to-model transformation and code generation for the OMNet++ simulator is completed in about 2 minutes. The simulation to evaluate the performance of one system configuration requires **54.5 seconds**.

Given the high complexity of the aforementioned model, this less-than-a-minute simulation-based performance analysis demonstrates the remarkable efficiency of CompDSE in evaluating a single design configuration. Such rapid evaluation is especially beneficial for industrial dCPS, where quick insights into the performance of complex machines are critical. Moreover, such rapid performance evaluation of a single design

Table 5.2: Time required for each step in the CompDSE workflow

Steps	Time Required
Adding tracepoints manually	about 5 minutes
Collecting execution traces automatically	4.33 minutes
Automated models derivation and integration	29.5 hours
Automated model transformation and code generation	1.82 minutes
Single point evaluation	54.5 seconds

configuration is crucial to enable fast exploration of large design spaces. For example, by integrating an automated search algorithm within CompDSE, such as a GA, and utilizing the rapid simulation-based performance analysis as a fitness function, we could significantly accelerate the exploration of large design spaces, making the CompDSE methodology even more scalable for future designs of complex dCPS.

5.4.2 Accuracy

In our experiment, the average processing duration per wafer (PDW) measured on the real ASML TWINSKAN machine is **50.548 seconds**, whereas the simulation of our derived system model of the same TWINSKAN machine indicates an average PDW of **49.183 seconds**, resulting in an error of 2.7%. This deviation reflects the reconstruction accuracy of the proposed modeling approach, indicating how closely the simulation reproduces the observed system behavior at the chosen level of abstraction. Together with the less-than-a-minute performance analysis time, this demonstrates that the model provides a faithful and computationally efficient abstraction of the system. Since the derived workload model preserves the structural dependencies and interaction patterns observed in the real execution, and the explored design alternatives modify only resource-related parameters within the same abstraction level, the validated baseline serves as a consistent reference for evaluating relative performance differences across configurations within the defined design space.

5.4.3 Exploration

Building on the validated baseline model, we conduct a small DSE experiment within the design space defined in Section 3.6. We first explore the effect of varying the number of cores in the main host (the Linux-based server) on the TWINSKAN system performance in terms of average PDW. The default configuration of the main host has sixteen cores. However, as shown in Figure 5.2, increasing the number of cores

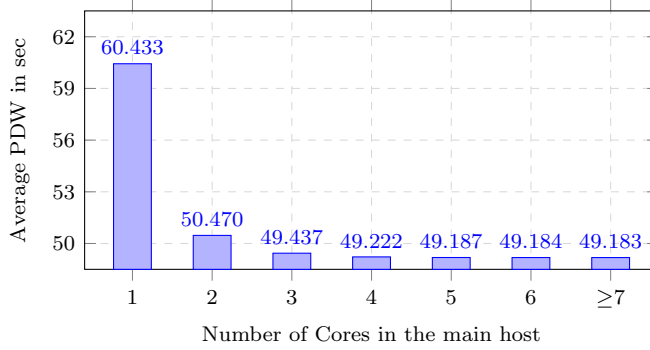


Figure 5.2: The impact of the number of cores on PDW

beyond seven does not lead to any significant changes in the average PDW. When we decrease the number of cores from seven to three, the average PDW does not increase significantly, i.e., the increase is in the range of milliseconds. Further decreasing the number of cores leads to a more substantial increase in the average PDW. The above findings suggest that the original design of the main host with sixteen cores in the ASML TWINSKAN machine might be an overestimation to avoid computing hardware resource bottlenecks. Thus, having a host with eight cores (half of the original design) could lower overall costs without compromising the machine performance, which could be considered in future system designs.

Next, we explore the effect of increasing the number of cores in the traced embedded hosts (the VxWorks-based real-time embedded systems) while keeping the main host with its default configuration. The results show that increasing the number of cores from one to four in these embedded hosts decreases the average PDW by 0.477 seconds. This small reduction indicates that while additional cores enable the system to handle more software processes in parallel, the overall impact on the average PDW is limited. This limitation is most likely due to the original design of the software infrastructure running in the ASML TWINSKAN machine where potential parallel processing in the software is not heavily exploited.

Finally, we examine the impact of varying the clock frequencies of the cores in the traced main and embedded hosts on the average PDW by conducting a series of simulations with different clock frequency settings, supported by the cores, across all hosts. The results, plotted in Figure 5.3, show the relationship between the average PDW and clock frequency changes relative to the base clock frequency setting of each host's multi-core CPU. The base frequency setting is denoted as 0 on the horizontal

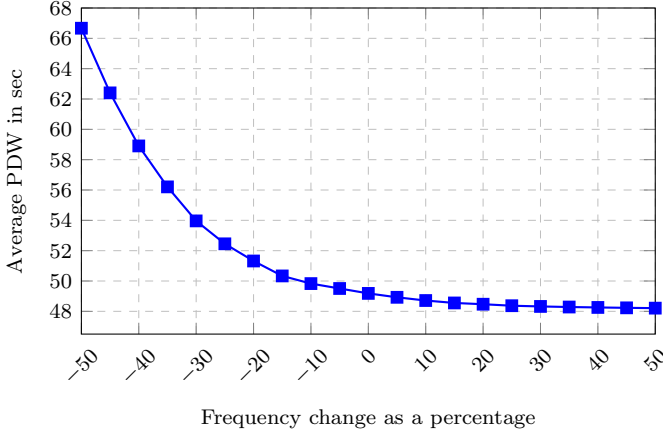


Figure 5.3: The impact of the clock frequency of cores on PDW

axis, while the average PDW is depicted on the vertical axis. The horizontal axis depicts the clock frequency changes as a percentage of the base clock frequency.

It can be seen in Figure 5.3 that increasing the clock frequency of all traced hosts' multi-core CPUs with 25% results in decrease of the average PDW from 49.183 to 48.371 seconds. However, increasing further the clock frequency yields only a marginal decrease of the average PDW, often within the millisecond range, as visualized by the curve's plateau in Figure 5.3. This plateau suggests that the performance gain in terms of PDW does not scale with the clock frequency increase when it is above 25%. The plateau indicates that the PDW is limited to around 48 seconds. This limitation arises from the performance of some untraced hosts reflected by the Δ parameter values.

The DSE experiments presented in this section focus on parameter variations of the traced platform configuration, specifically changes in computational resources such as the number of cores and clock frequencies. Although the design space defined in Section 3.6 also allows structural platform modifications, including the addition of hosts or changes in topology, such structural variations were not part of the present validation.

5.5 Conclusion

This chapter evaluates the CompDSE methodology using a realistic industrial case study, namely the ASML TWINSKAN lithography machine. The results show that the models, automatically derived by CompDSE, closely reproduce the observed behavior of

the real system under the same architectural configuration. In particular, the simulated average wafer processing time differs by 2.7% from the measured system processing time, which reflects the reconstruction accuracy of the proposed modeling approach at the adopted level of abstraction. Despite the complexity and scale of the model, each design point could be simulated in under one minute, demonstrating that the methodology remains computationally efficient. Building on the validated baseline model, we conduct a small DSE experiment within the defined design space to assess how architectural changes affect the system performance. The experiment demonstrates that reducing the number of cores can lower the system cost without significantly affecting the performance, while adjusting clock frequencies allows designers to observe how much performance gain can be achieved within the limitations of the physical environment. These findings highlight the applicability of CompDSE in supporting early-stage architectural design decisions.

Although the exploration in this chapter is conducted manually, the CompDSE methodology is well-suited for integration with search algorithms, such as GA, to further enhance automation and enable efficient large-scale optimization of complex dCPS.

