PREPRINT - ©2024 IEEE - 2024 29th Asia and South Pacific Design Automation Conference (ASP-DAC)
DOI: 10.1109/ASP-DAC58780.2024.10473967

The Optimal Quantum of Temporal Decoupling

*Niko Zurstraßen, *Ruben Brandhofer, *José Cubero-Cascante,
*Nils Bosbach, †Lukas Jünger, *Rainer Leupers

*RWTH Aachen University, Institute for Communication Technologies and Embedded Systems
†MachineWare GmbH, Aachen, Germany

***Abstract*—Virtual Platforms (VPs) and Full System Simulators (FSSs) are among the fundamental tools of modern Multiprocessor System on A Chip (MPSoC) development. In the last two decades, the execution speed of these simulations did not grow at the same rate as the complexity of the systems to be simulated, creating a need for faster simulation techniques. A popular approach is *temporal decoupling* (TD), in which parts of the simulation are not synchronized with the rest of the system for a time called *quantum*. A high quantum is beneficial for simulation performance due to fewer synchronization/context switches. Yet, it also increases the probability of causality errors, leading to inaccuracies. Thus, most users of TD simulations face the question: Which quantum offers the optimal compromise between accuracy and performance? In practice and literature, the quantum is usually chosen based on empirical knowledge. This approach can achieve adequate performance/accuracy, but it lacks proper reasoning. In this work, we address this shortcoming by providing analytical estimations and deeper insights into the effects of Temporal Decoupling (TD). Additionally, we verify the proposed models using TD simulations in SystemC and gem5.**

***Index Terms*—Temporal Decoupling, SystemC, gem5, Full-System Simulation, Quantum**

I. Introduction

Virtual twins of compute systems, so-called Virtual Platforms (VPs) or Full System Simulators (FSSs), are versatile tools used in the state-of-the-art development of hardware, software, and their co-design. Following the trend of Moore's Law, the complexity of compute systems is growing exponentially, leading to increasingly complex VPs. However, since the execution speed of the often single-threaded VPs has hardly benefited from new processor generations in the last two decades, innovative methods are needed to close this gap of complexity and simulation performance.

An established method to accelerate simulations is TD. With TD, simulation processes do not synchronize with the simulation kernel for up to $t_{\Delta q}$ (the so-called *quantum*). This leads to a fewer synchronizations/context switches, which in turn increases the simulator's performance. Moreover, the relaxed constraints on chronological event ordering are a key enabler for parallel simulations, allowing for even higher speedups. However, the relaxed constraints can lead to a chronologically incorrect execution order of events. This negatively impacts the simulation's accuracy, or in the worst case, flaws the functionality of the simulated system. Therefore, small quanta increase the simulation's accuracy, while maximum performance is attained by large quanta. Since simulations are supposed to be fast and accurate, one faces the dilemma of finding a quantum that yields a bearable inaccuracy with significant speedups. In many works, the optimal quantum is determined by empirical observations and is not further elaborated:

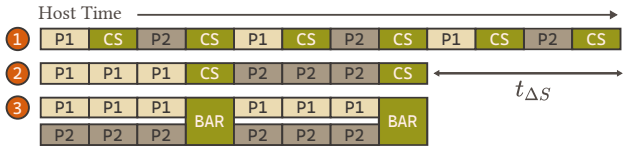

Fig. 1. Depicting the principle of temporal decoupling.

J. Engblom [11]: "Time quantum lengths of 10k to 1M cycles are needed to maximize VP performance. Most of the time, software functionality and correctness are unaffected by TD, and the default should be to use long time quanta."

Ryckbosch et al. [21]: "We set the simulation window to 10ms and the simulation quantum to 100ms in all of our experiments. We experimentally evaluated different values for the simulation window and quantum, and we found the above values to be effective."

In this work, we address this shortcoming by providing analytical models for the performance estimation of TD simulations. We show that speedup and accuracy can be precisely determined with simple formulas. However, the gist of our analytical model is not its quantitative assessment, but its statement about how the quantum is related to the law of diminishing returns. This should guide the expectations of VP users into the right direction, and help to understand why a larger quantum does not necessarily equal higher performance.

II. Background

A. Temporal Decoupling (TD)

TD allows individual simulation processes to advance in their local time by a simulation time $t_{\Delta q}$ from the rest of the system. This principle is illustrated in Fig. 1 with Case 1 showing the execution of a sequential simulation without TD. The depicted system comprises two processes (P1, P2) that execute events alternately. For example, this could be a dual-core system where each event corresponds to the execution of an instruction with a constant time, e.g. 1ns. To ensure correct chronological execution, the simulation kernel has to switch processes after each instruction (Case 1). TD reduces the number of context switches by allowing each simulation process to run ahead of global time by up to a quantum $t_{\Delta q}$ (Case 2). Still assuming the dual-core example with 1 GHz clock speed, a quantum of $t_{\Delta q} = 3ns$ would correspond to an uninterrupted execution of 3 instructions. This results in 2 context switches instead of 5, reducing the simulation time by $t_{\Delta S}$.

PREPRINT - ©2024 IEEE - 2024 29th Asia and South Pacific Design Automation Conference (ASP-DAC)
DOI: 10.1109/ASP-DAC58780.2024.10473967

TD can also be used to enable parallel simulations, as shown in Case ③. Here, each process runs in its own thread with its own local time and must synchronize with the other threads at a barrier every $t_{\Delta q}$. Generally, such a method is also called Parallel Discrete Event Simulation (PDES), which is often subdivided into conservative and optimistic approaches. Optimistic PDES rectifies causality errors through rollbacks, while conservative PDES prevents them outright. Since the presented approach allows causality errors, it does not fall into either category.

In both sequential and parallel cases, TD can lead to an incorrect chronological execution order. However, compute systems are remarkably resilient to causality errors. Whether a CPU is a few cycles ahead or if a timer interrupt occurs a few microseconds too late, is irrelevant to the functionality of most systems. Thus, often only the timing accuracy of the target system is minimally affected. Nevertheless, for large quanta, i.g. in the order of milliseconds or greater, functionality can be affected. To counteract these negative effects, the quantum can also be terminated early in sequential TD simulations. For example, if a CPU executes a wait-for-interrupt (WFI) instruction, it is foreseeable that the CPU will not perform any meaningful task until the quantum ends, as the expected interrupt will originate from another simulation thread. Hence, the current quantum can be terminated early.

III. Related Work

The principles of temporal decoupling (TD) were first introduced in 1969 by Fuji et al. [13], who call their method *slice-based*, but already introduce the term *quantum*. Today's prevalent term *temporal decoupling* was coined nearly 40 years later by the release of SystemC TLM-2.0 standard [5], [6] in 2008. Even though TD is of practical relevance, the question of the optimal quantum is still a relatively uncharted field of research.

There are several works tackling the inaccuracy of TD. For example, Jung et al. propose *speculative temporal decoupling* that allows for a completely error-free simulation by re-simulating erroneous sections with tighter synchronization. A similar approach is used by [10], [15]. Glaser et al. [14] present a control-theory-based approach called *predictive temporal decoupling*. In their work, a Wiener filter is used to predict events and dynamically adjust the length of the quantum to minimize inaccuracy. In order to mitigate the diminishing performance of dynamic TD, Jünger et al. [16] show how the relation between events can be analyzed to allow for greater quantum sizes without decreasing the simulation's accuracy. In the context of parallel TD simulation, a popular approach is to set the quantum to a communication channel's minimum delay [9], [19], [22]. This allows for simulations completely free of errors, but drastically limits the use-cases.

To the best of our knowledge, the only works combining analytical models and TD simulation were published by Falsafi et al. [12] and Over et al. [20]. Falsafi et al. present an analytical performance model for the *Wisconsin Wind Tunnel Simulator* to determine the financial cost-effectiveness of parallel simulations. Although the work by Falsafi et al. builds on a similar foundation as ours, it leaves the question of the optimal quantum untouched. In a similar fashion, Over et al. [20] employ analytical models to compare two parallelization methods. But again, the model is rather a byproduct, and the question for the optimal quantum remains unanswered.

IV. Methods

A. Analytical Speedup Models

In this section, we develop mathematical models to predict simulation speed and accuracy as a dependency of the quantum. Since the principle of parallel TD simulations differs from their sequential counterpart, two separate analytical models for the performance estimation are introduced.

1) Temporally-Decoupled Parallel Simulation: The first model describes the speedup of a TD parallel simulation. As explained in Section II and shown in Fig. 1 Case ③, this kind of simulation leverages the host system's multi-threading capability by executing parts of the simulation in parallel. For example, with a simulated multi-core system, each target CPU can be executed in its own thread. Since this is a common use case, such systems will be assumed in the following. Furthermore, assuming the system's simulation and its workload are perfectly parallelizable, and there are no synchronization barriers, the following formula for the speedup S holds valid:

$$S(N) = T/(T/N) = N \tag{1}$$

With T being the host time required for a sequential simulation and N being the number of threads. As seen in Equation 1, the speedup grows linearly with the number of simulation threads. For most parallel applications, a speedup of N represents an upper estimate, as there is additional overhead due to the synchronization of threads. In a synchronous parallel simulation, this overhead is imposed by periodic barrier events. We model this by adding a relative workload $O_b(t_{\Delta q}, N)$, which depends on the length of the quantum $t_{\Delta q}$ and the number of threads N:

$$S(N, t_{\Delta q}) = \left(1/N + O_b(t_{\Delta q}, N)/N\right)^{-1} \tag{2}$$

Since the overhead is inversely proportional to the quantum ($O_b(t_{\Delta q}, N) = O'_b(N)/t_{\Delta q}$), the formula can be refined to:

$$S(N, t_{\Delta q}) = \left(1/N + O'_b(N)/(N \cdot t_{\Delta q})\right)^{-1} \tag{3}$$

On modern operating systems, we observed that the timing overhead imposed by barriers grows linearly with the number of threads. This was found to be independent of the operating system, hardware, or barrier type. Consequently, $O'_b(N) = O''_b \cdot N$, which leads to:

$$S(N, t_{\Delta q}) = \left(1/N + O''_b/t_{\Delta q}\right)^{-1} \tag{4}$$

Ultimately, the factor O''_b remains, representing a measure for the overhead of a synchronization independent of the number of cores. By rearranging Equation 4, O''_b can be obtained as:

$$O''_b = t_{\Delta q} \cdot \frac{N - S(N, t_{\Delta q})}{S(N, t_{\Delta q}) \cdot N} = t_{\Delta q} \cdot \frac{t_N(t_{\Delta q}) - t_{seq}/N}{t_{seq}} \tag{5}$$

PREPRINT - ©2024 IEEE - 2024 29th Asia and South Pacific Design Automation Conference (ASP-DAC)
DOI: 10.1109/ASP-DAC58780.2024.10473967

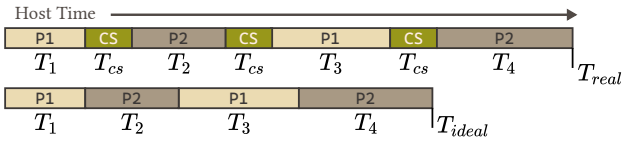

Fig. 2. Example timings for a sequential temporally-decoupled simulation.

With $t_N(t_{\Delta q})$ being the host execution time of a parallel simulation using N cores and a quantum of $t_{\Delta q}$. The time for the sequential reference is given by t_{seq}. Hence, two simulation runs are required to estimate O_b''. Since the parameter O_b'' establishes a link between workload and synchronization, it must be determined for each workload individually. In practice, fluctuations in O_b'' are so small that sufficiently accurate statements can be made with just one measurement. With par-gem5, for example, all measurements for O_b'' were in the range of 2-6ns. The obtained formula can be further refined by accounting for sequential parts of the simulation. For example, there could be a global state in the simulation, which must be protected by serial access. But also the parallelism of the executed target workload affects the speedup. A simulated single-core Dhrystone benchmark will not benefit from a parallel simulation, as the workload is inherently sequential. To also model the impact of parallelism, we extended the formula by *Amdahl's Law* [7] to:

$$S = \left(\frac{p}{N} + 1 - p + \frac{O_b''}{t_{\Delta q}} \right)^{-1} \tag{6}$$

Amdahl's Law introduces a factor $p \in [0, 1]$ that indicates to what extent the workload is parallelizable. Since p is an unknown variable, it must be determined, for example, by using a system of linear equations derived from Equation 4. In the following, we use the least error squares method to determine the parameters O_b'' and p.

2) Temporally-Decoupled Sequential Simulation: In this subsection, we present an analytical model for TD sequential simulations as implemented in the SystemC standard. With sequential TD, the speedup is attained by reducing the number of the simulator's context switches. Hence, for a perfect simulation without any context switches, the ideal time (T_{ideal}) is given by the sum of the time of all simulation segments T_i (see Fig. 2): $T_{ideal} = \sum_{i=1}^{K} T_i$. Practically, the context switches between the individual simulation segments lead to a relative overhead O_c:

$$T_{real} = T_{ideal} \cdot (1 + O_c) \tag{7}$$

Similar to the parallel simulation, this overhead is inversely proportional to the chosen quantum. Consequently:

$$T_{real} = T_{ideal} \cdot (1 + O_c'/t_{\Delta q}) \tag{8}$$

This also assumes the chosen quantum is larger than the average event distance. For most cases in real-world VPs, this assumption holds valid. The factor O_c' can be determined by running two reference simulations for a given VP:

$$\frac{T_{real}(t_{\Delta q1})}{T_{real}(t_{\Delta q2})} = \frac{1 + \frac{O'}{t_{\Delta q1}}}{1 + \frac{O'}{t_{\Delta q2}}} \Rightarrow O_c' = \frac{T(t_{\Delta q_1}) - T(t_{\Delta q_2})}{\frac{T(t_{\Delta q_2})}{t_{\Delta q_1}} - \frac{T(t_{\Delta q_1})}{t_{\Delta q_2}}} \tag{9}$$

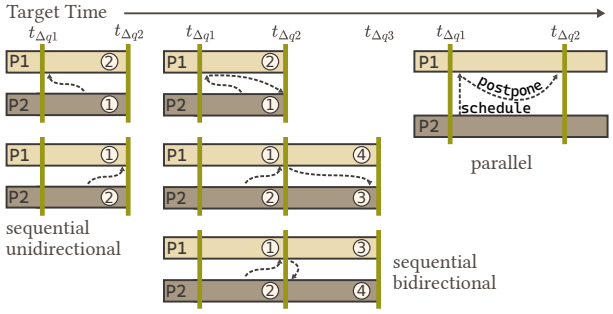

Fig. 3. Types of timing errors in TD simulations.

To accurately determine the factor O_c', we recommend choosing low quanta, for which the context switching time is a significant fraction of the total simulation time. Alternatively, the factor O_c' can be determined by multiple reference simulations and curve fitting. Ultimately, the speedup can be derived as:

$$S(t_{\Delta q}) = \frac{T_{ideal}}{T_{real}} = \frac{t_{\Delta q}}{t_{\Delta q} + O_c'} \tag{10}$$

The formula always yields values smaller than 1, since we assume that TD does not accelerate the simulation, but reduces performance-degrading environmental effects.

B. Analytical Inaccuracy Models

Contrary to the clearly defined speedup, the term "inaccuracy" can be understood in various ways. First of all, inaccuracy can be categorized into qualitative and quantitative aspects. Qualitative inaccuracy includes all observations that cannot be expressed as a metric and lead to changed simulation semantics. E.g., if TD leads to a program's crash, the effect can be characterized as qualitative.

Quantitative inaccuracy, on the other hand, can be meaningfully captured in numbers. For example, it can be the inaccuracy of cache hit rates, interrupt timings, or simulation time. In the following, we chose the target simulation time as a representative measure of accuracy, as it is influenced by other metrics and can be found in most simulators.

1) Quantitative Inaccuracy: Parallel Simulation: Ultimately, any kind of inaccuracy for TD simulations is caused by an incorrect execution order of events. In other words, from the point of view of a process, events occur too early or too late. This is exemplified in Fig. 3. In the case of a par-gem5, events beyond thread boundaries are always postponed to the next quantum boundary if the event time is prior to the next barrier. Therefore, the simulation duration in target time is extended if the event is on the critical path. To analytically determine the length of extension, we first assume that most events are scheduled without delay. This assumption is consistent with our observations, as most events, such as cache snoops or SEV instructions, are scheduled quasi-instantaneously. Only a few events, like timer interrupts, are scheduled far into the future. Additionally, we assume that cross-scheduled events occur stochastically independent of each other and always lie on the critical path. Under this assumption, the occurrence of a first cross-scheduled event within a quantum can be modeled

PREPRINT - ©2024 IEEE - 2024 29th Asia and South Pacific Design Automation Conference (ASP-DAC)
DOI: 10.1109/ASP-DAC58780.2024.10473967

with the Poisson distribution, leading to an average extension of the simulation of t_d per quantum:

$$\begin{aligned} t_d &= t_{\Delta q} - E(X|X \leq t_{\Delta q})P(X < t_{\Delta q}) \\ &\quad - t_{\Delta q}P(X > t_{\Delta q}) \\ &= t_{\Delta q} - \int_0^{t_{\Delta q}} rte^{-rt}\,dt - \int_{t_{\Delta q}}^{\infty} rt_{\Delta q}e^{-rt}\,dt \\ &= t_{\Delta q} - (1 - e^{-rt_{\Delta q}})/r \end{aligned} \tag{11}$$

This results in the relative timing inaccuracy of:

$$I = \frac{t_{\Delta q}}{t_{\Delta q} - t_d} - 1 = \frac{r \cdot t_{\Delta q}}{1 - e^{-rt_{\Delta q}}} - 1 \approx r \cdot t_{\Delta q} \tag{12}$$

With r being the rate of cross-scheduled events per time unit. As seen from Equation 12, the relative timing inaccuracy is described by a hockey stick curve. Initially, the relative inaccuracy grows exponentially for small quanta and then transitions to a linear growth $r \cdot t_{\Delta q}$, which also acts as an upper bound. This is in stark contrast to the derived speedup model. While the achievable speedup eventually saturates, the inaccuracy continues to increase indefinitely. This underlines why the choice of the optimal quantum is so essential.

2) Quantitative Inaccuracy: Sequential Simulation: While errors in parallel simulations are primarily caused by the postpone process, the situation for sequential TD simulation is more complicated. Considering only unidirectional communication between two processes (P1, P2) as shown in Fig. 3, two types of errors can occur. If process P2 is executed first and sends a message to process P1 within the quantum, P1 will have received this message right at the beginning of its quantum. Thus, the message will be received too early on average by $t_{\Delta q}/2$. On the other hand, if P1 is executed first, the message will not reach this process until the next quantum. Consequently, this message will be received $t_{\Delta q}/2$ too late on average. If both cases are considered equally probable, and if the characteristics of the executed target software are neglected, there will be no time deviations on average.

For a bidirectional communication the situation is different (see Fig. 3). In the example shown, P2 wants to send a message to process P1, which is answered by the latter. For simplicity, we assume that the target time required for sending and receiving a message is minimal compared to the length of the quantum. Again different cases arise depending on the execution sequences. However, for all cases the simulation time is extended by either $t_{\Delta q}/2$ or $3t_{\Delta q}/2$. Therefore, on statistical average, the target simulation time for sequential TD simulation is extended. Using a similar approach as for parallel simulations, the inaccuracy can be approximated by a linear term. Thus, it can be stated that in both the parallel and sequential case, the relative timing inaccuracy depends linearly on the quantum. Without specifying the linear factor in particular, this yields the following equation for the timing inaccuracy I:

$$I = \alpha \cdot t_{\Delta q} \tag{13}$$

The factor α can be determined by two reference simulations or curve fitting.

V. Results and Discussion

To substantiate the statements and accuracy of our model, validations with reference simulations are a necessity. All following simulations were executed on an *AMD Ryzen 3990x* (64 physical cores/128 logical cores) host system.

A. Parallel TD

For the parallel TD simulations we conducted experiments using a parallel fork of gem5 [8], called par-gem5 [23]. The principles of par-gem5 follow the background and formulas explained in Subsection II-A and Subsection IV-A2. As a target, a {16,32,64}-core Simple-CPU ARM system clocked at 1 GHz with a multi-level cache hierarchy was simulated. Using this setup, we executed a bare-metal bubble-sort application, NPB IS.W [3], and STREAM [18]. All benchmarks were executed with Ubuntu as the target's operating system, while the bubble-sort application ran bare metal. In addition, the benchmarks are multi-core applications that distribute their workload across all cores of the target system. We refrained from running single-threaded benchmarks, since the execution of sequential workloads in parallel TD simulations is not associated with significant speedups. The results of our experiments can be found in Fig. 4. It can be observed that both speedup and accuracy follow the general trend of the analytical model. The achievable speedups vary from 1 (STREAM) up to 20× for the bare-metal bubble sort benchmark, which is reflected in the respective parallelization factor. However, all benchmarks share a factor O_b'' in the same order of magnitude, which is why the speedups saturate around a quantum of 500ns.

The relative inaccuracy in simulation time shows a quasi-linear growth, as predicted. Depending on the benchmark, the slope of the inaccuracy line differs by a factor of more than 20×. In contrast to the speedup, however, a significantly larger scatter around the predicted value can be observed here (see STREAM). For STREAM, we attribute the deviations to the periodic behavior of benchmark-internal functions. Unlike initially assumed, cross-scheduled events do not necessarily follow a Poisson process, but show a periodic behavior. Although the inaccuracy also increases linearly for these in the arithmetic mean, particularly advantageous or disadvantageous sweet spots can be hit with certain quanta.

B. Sequential TD

As a reference implementation for sequential TD simulation, we used the open-source ARMv8 TLM-2.0-based [6] VP *avp64* [1] and a RISC-V VP based on MachineWare's *SIM-V* [17]. To allow comparison with par-gem5, the CPUs of both VPs are clocked at 1 GHz. As benchmarks we chose Dhrystone, NPB, STREAM, and an operating system boot. All benchmarks were executed on a buildroot-configured Linux system. The results in Fig. 4 show a similar trend of diminishing speedup returns and linearly increasing inaccuracy. Compared to gem5 the speedup begins to saturate at values greater than 10µs. This can be attributed to the efficient dynamic binary translation backend of the simulators and the relatively

PREPRINT - ©2024 IEEE - 2024 29th Asia and South Pacific Design Automation Conference (ASP-DAC)
DOI: 10.1109/ASP-DAC58780.2024.10473967

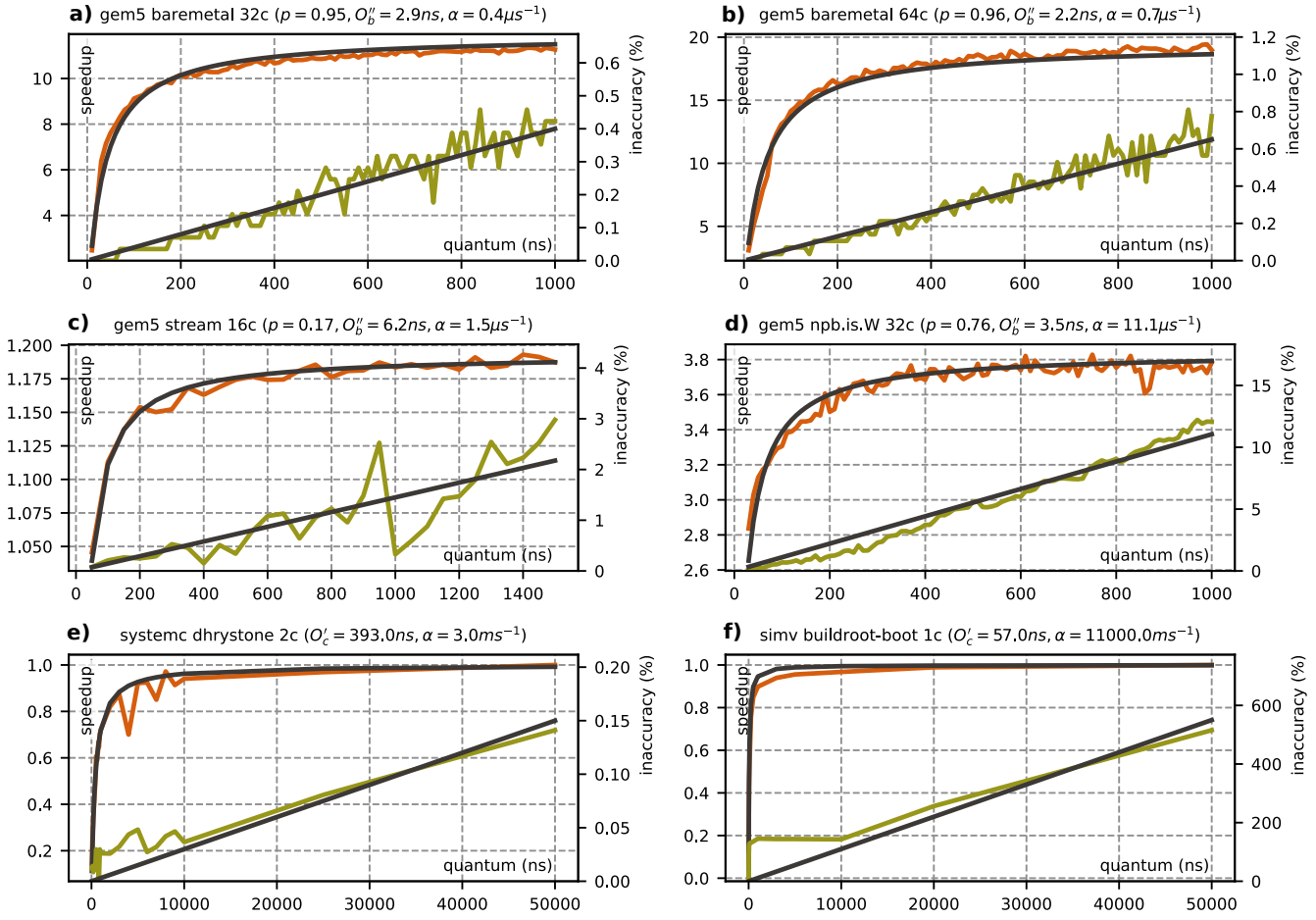

Fig. 4. Measurement vs. estimation for speedup (orange) and target time inaccuracy (green) using various benchmarks and platforms.

TD	default	
[0.000385]	[0.000385]	Mount-cache hash table entries: 32768 [...]
[0.000396]	[0.000396]	Mountpoint-cache hash table entries: [...]
[422.828066]	[0.024140]	ASID allocator initialised with 128 entries
[3495.801687]	[0.032140]	Hierarchical SRCU implementation.
[845.656091]	[0.048162]	smp: Bringing up secondary CPUs ...
[5877.941435]	[0.080218]	Detected PIPT-Icache on CPU1

Fig. 5. Linux boot timestamps of a TD and a default simulation in gem5.

huge overhead of context switches, which is also reflected in the factor O_c'. For example, the Dhrystone benchmark attains a value of $O_c' = 393ns$, which means the execution of 393 cycles takes just as long as one SystemC context switch. This example shows why TD has become a staple of Electronic System Level (ESL) simulations. Simulated with a quantum of 10µs, Dhrystone needs about 10 minutes of host simulation time. Without TD, the simulation would require more than 2 days. In contrast to the speedup, the inaccuracy behaves unpredictably at first and only changes to a linear growth from a quantum of 10µs. The CPU is particularly responsible for low quanta, which can end its quantum earlier in sequential TD (see Section II). Thus, an increasing quantum does not necessarily lead to increasing inaccuracy, as can be seen in the example of the *SIM-V* boot process (Fig. 4 f).

C. Qualitative Inaccuracy

During the execution of our experiments, not only quantitative inaccuracy in the form of timing errors occurred, but also qualitative inaccuracy could be observed. This subsection discusses the observed deviations in detail as the effects were not always obvious, yet significantly changed the meaning of the simulation, In our simulations, we were always able to attribute qualitative inaccuracy to either access from different local time domains or delay of communication. The former error is prevalent for simulation objects, which require simulation timestamps for their functionality. An example of this is the gem5 implementation of the ARM virtual count `CNTVCT_EL0` register. When fetching the value of the register, its current value is determined by the time difference between the current and the last access. In TD simulations, the last access may have a higher timestamp, resulting in a negative delta. This delta is stored in an unsigned integer, leading to corrupted values. These values can be observed during a Linux boot, where the timestamps of boot messages are determined by the `CNTVCT_EL0` register. If `CNTVCT_EL0` is corrupted, so are the printed timestamps (see Fig. 5). A similar error was also observed by Engblom [11], who proposes a restriction to

PREPRINT - ©2024 IEEE - 2024 29th Asia and South Pacific Design Automation Conference (ASP-DAC)
DOI: 10.1109/ASP-DAC58780.2024.10473967

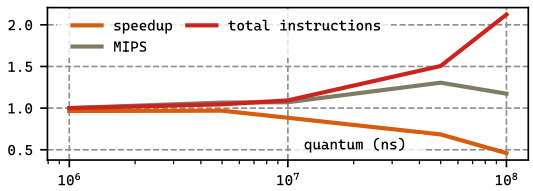

Fig. 6. Comparing normalized MIPS, speedup, and instructions for the NPB IS benchmark running on the avp64 SystemC VP.

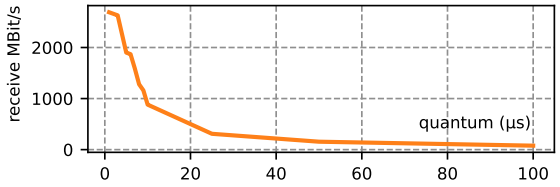

Fig. 7. Executing the *iperf3* benchmark using the VP as the server and the host system as a client. The throughput uses the simulation time as a reference.

deltas greater than or equal to zero.

The second type of observed error arises from delayed communication between simulation objects. In the case of the multi-threaded NPB benchmark, we observed that the synchronization of threads was delayed by TD. Moreover, while threads wait for their synchronization, they dwell in a spin loop, executing NOP instructions. For large quanta, this leads to an interesting effect: The total number of instructions executed increases, causing the speedup measured in host execution time to decrease. However, the speedup of the simulator measured in MIPS stagnates or even increases since NOPs are easy to simulate. As shown in Fig. 6, first effects are already visible at $t_{\Delta q} > 1ms$. At $t_{\Delta q} > 100ms$, more than half of the time is spent in spin loops. We therefore advise to not use MIPS as performance metric for TD simulations.

In addition to the effects on simulation performance, throughput or functionality of peripherals can also be affected by delayed communication. As an example, we executed the *iperf3* [2] benchmark in *avp64* with the VP as a server and the host system as a client. In our configuration, the benchmark determines the maximum throughput of a TCP-based connection between a server and a client. As shown in Fig. 7, the throughput rapidly decreases from 2690 Mbit/s at $t_{\Delta q} = 1s$ to 77 MBit/s at $t_{\Delta q} = 100s$. This performance drop can be explained by the implementation of the OpenCores Ethernet device *ETHOC* [4] which is used in *avp64*. The device uses one thread each for sending and receiving Ethernet frames, and each of these threads is executed only once per quantum. Thus, only one Ethernet frame can be received per quantum, which limits the maximum achievable throughput. Ultimately, this can affect the data rate to such an extent that timeouts of the network driver watchdog occur.

VI. Conclusion

In this paper, we showed how analytical models can be used to estimate the performance and accuracy of parallel and sequential TD simulations. Rather than improving the model's accuracy by adding complexity, we tried to capture the gist of TD in a few simple formulas. Nevertheless, the model can estimate with sufficient accuracy, which we have verified with case studies conducted with SystemC and par-gem5. The general rule of thumb that with larger quanta the speedup increases and the accuracy decreases could be confirmed by our model and measurements. Yet, our model allows to quantify this statement further. The speedup of the simulation as a function of the quantum behaves similarly to a saturating sigmoidal curve, while the simulation inaccuracy, on the other hand, grows linearly and indefinitely. This means, the achievable speedup has an upper bound, and above a certain point, increasing the quantum mainly deteriorates the simulation's accuracy. Ultimately, the optimal compromise between performance and accuracy is still determined by the user, however, our models are of substantial value because they provide guide-rails to focus discussion and hone expectations for this decision.

References

[1] "ARMv8 Virtual Platform (AVP64)," https://github.com/aut0/avp64, accessed: 2022-01-09.
[2] "iperf3 benchmark," https://software.es.net/iperf/, accessed: 2022-11-07.
[3] "NAS Parallel Benchmarks," https://www.nas.nasa.gov/software/npb.html, accessed: Accessed: 2022-01-08.
[4] "OpenCores Ethernet MAC 10/100 Mbps," https://opencores.org/projects/ethmac, accessed: 2022-11-07.
[5] "OSCI TLM-2.0 Language Reference Manual," https://www.accellera.org/images/downloads/standards/systemc/TLM_2_0_LRM.pdf, accessed: 2022-08-10.
[6] "IEEE Standard for Standard SystemC Language Reference Manual," *IEEE Std 1666-2011 (Revision of IEEE Std 1666-2005)*, 2012.
[7] G. M. Amdahl, "Validity of the Single Processor Approach to Achieving Large Scale Computing Capabilities," in *Proceedings of the April 18-20, 1967, Spring Joint Computer Conference*, 1967.
[8] N. Binkert *et al.*, "The Gem5 Simulator," *SIGARCH Comput. Archit. News*, 2011.
[9] D. Burger *et al.*, "Accuracy vs. performance in parallel simulation of interconnection networks," in *Proceedings of 9th International Parallel Processing Symposium*, 1995, pp. 22–31.
[10] G. Busnot *et al.*, "Standard-compliant Parallel SystemC simulation of Loosely-Timed Transaction Level Models," in *ASP-DAC*, 2020.
[11] J. Engblom, "Temporal Decoupling-Are 'Fast'and 'Correct'Mutually Exclusive?" in *DVCon Europe*, 2018.
[12] B. Falsafi *et al.*, "Cost/Performance of a Parallel Computer Simulator," in *Proceedings of the Eighth Workshop on Parallel and Distributed Simulation*, ser. PADS '94, 1994.
[13] K. Fuchi *et al.*, "A Program Simulator by Partial Interpretation," in *Proceedings of the Second Symposium on Operating Systems Principles*, ser. SOSP '69, 1969.
[14] G. Glaser *et al.*, "Temporal decoupling with error-bounded predictive quantum control," in *FDL*, 2015.
[15] D. R. Jefferson, "Virtual Time," *ACM Trans. Program. Lang. Syst.*, 1985.
[16] L. Jünger *et al.*, "Optimizing Temporal Decoupling using Event Relevance," in *ASP-DAC*, 2021.
[17] ——, "SIM-V: Fast, Parallel RISC-V Simulation for Rapid Software Verification," *DVCON Europe*, 2022.
[18] J. McCalpin, "Memory bandwidth and machine balance in high performance computers," *IEEE Technical Committee on Computer Architecture Newsletter*, pp. 19–25, 12 1995.
[19] A. Mohammad *et al.*, "dist-gem5: Distributed simulation of computer clusters," in *IEEE ISPASS*, 2017.
[20] A. Over *et al.*, "A Comparison of Two Approaches to Parallel Simulation of Multiprocessors," *Performance Analysis of Systems and Software, IEEE International Symmposium on*, vol. 0, pp. 12–22, 04 2007.
[21] F. Ryckbosch *et al.*, "VSim: Simulating Multi-Server Setups at near Native Hardware Speed," *ACM Trans. Archit. Code Optim.*, jan 2012.
[22] J. H. Weinstock *et al.*, "SystemC-link: Parallel SystemC simulation using time-decoupled segments," in *DATE*, 2016.
[23] N. Zurstraßen *et al.*, "par-gem5: Parallelizing gem5's Atomic Mode," to appear in DATE, 2023.