GPU-based Architectures and their Benefit for Accurate and Efficient Wireless Network Simulations

Philipp Andelfinger[*], Jens Mittag[†], Hannes Hartenstein[*†]
[*]Steinbuch Centre for Computing, Karlsruhe Institute of Technology, Germany,
Email: {philipp.andelfinger, hannes.hartenstein}@kit.edu
[†]Institute of Telematics, Karlsruhe Institute of Technology, Germany, Email: jens.mittag@kit.edu

***Abstract*—In recent years, a trend towards the usage of physical layer models with increased accuracy can be observed within the wireless network community. This trend has several reasons. The consideration of signals – instead of packets – as the smallest unit of a wireless network simulation enables the ability to reflect complex radio propagation characteristics properly, and to study novel PHY/MAC/NET cross-layer optimizations that were not directly possible before, e.g. cognitive radio networks and interference cancelation. Yet, there is a price to pay for the increase of accuracy, namely a significant decrease of runtime performance due to computationally expensive signal processing. In this paper we study whether this price can be reduced – or even eliminated – if GPU-based signal processing is employed. In particular, we present and discuss four different architectures that can be used to exploit GPU-based signal processing in discrete event-based simulations. Our evaluation shows that the runtime costs can not be cut down completely, but significant speedups can be expected compared to a non GPU-based solution.**

I. Introduction

Within the past few years, a tendency towards the usage of more accurate representations for the lower layers can be observed in the wireless networking community. What started by means of experimentation platforms and testbeds that are based on software-defined radios [1] or channel emulation techniques [2] has recently also been adopted to wireless network simulations [3]. The motivation to increase the accuracy of the lower layers, in particular of the physical layer and below, is twofold. First, it allows to reflect the characteristics of complex radio propagation conditions properly, i.e. time- and frequency selective channels, and second, it provides the ability to study the feasibility and performance of emerging new concepts such as cognitive radio networks, interference cancelation or other cross-layer optimizations. For an in-depth discussion of the motivation to increase the accuracy and further examples we refer to [3].

In comparison to the abstraction of packets, as done by traditional network simulators which typically ignore individual bits and consider the packet as their smallest unit, an accurate representation explicitly considers individual bits and their modulation as complex time samples, cf. Figure 1. As illustrated, the signal-level consideration describes a packet in much more detail and can require a sequence of up to thousands of complex time samples to describe a packet completely. For the rest of this paper, we term a simulation

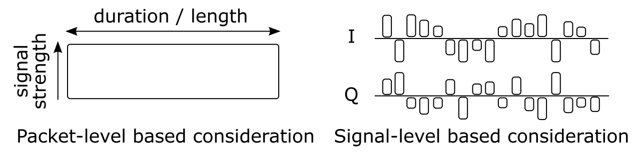
duration / length
signal strength
I
Q
Packet-level based consideration
Signal-level based consideration

Fig. 1. Comparison of packet-level and signal-level modeling of a data frame.

that treats each packet as a collection of bits and corresponding signals: *accurate wireless network simulation*.

Although there are many reasons to opt for accurate wireless network simulations, there is a price to pay. As shown in [3], the runtime performance of such simulations is reduced by up to four or five orders of magnitude due to a data intensive and computationally expensive signal processing. Hence, there is a trade-off between accuracy and runtime performance and one should ask oneself twice whether one needs the increase of accuracy or not. Considering that an increased accuracy is desired or required, the question is raised whether the necessary costs can be reduced. In this paper, we therefore study whether this price can be reduced – or even eliminated – if data intensive signal processing algorithms are parallelized and ported to GPUs. In particular, we present and compare four different architectures for a GPU-accelerated simulation of accurate wireless networks. All four architectures aim to decrease the overhead that exists due to data transfers from/to GPU and aim to increase the amount of data to be processed in parallel. While the most simple and less effective architecture does not require any changes to the runtime logic of a discrete event-based network simulator, our more effective proposals require changes and employ aggregation of independent simulation events and/or memory reuse on GPUs.

The rest of the paper is structured as follows. First, we cover related work in Section II and give a brief introduction to accurate wireless network simulation in Section III. In Section IV we then present performance results of GPU-based signal processing, as well as present, compare and discuss four GPU-based architectures for the simulation of accurate wireless networks. Section V concludes this paper.

II. Related Work

Parallel simulation using GPUs has been approached in various ways in literature. Xu et al. [4] proposed a simulation platform utilizing GPUs for signal processing tasks and evaluated the platform with regard to runtime performance. The platform integrates CPU-based simulator logic and GPU-based processing for computationally intensive tasks, corresponding to the "hybrid" approaches presented later in this paper. Xu et al. showed that GPU-based processing has clear performance benefits when maximizing the amount of input data per GPU work cycle. However, they state that algorithms used for encoding and decoding of packets in communication systems are not suitable to exploit the parallel processing capabilities of GPUs. Furthermore, it seems that Xu et al. only considered multiple events within the context of one node for aggregation, and not within the context of multiple network nodes.

In 2006, Perumalla et al. [5] presented a first study on whether GPU-based processing can improve the performance of simulators with fixed time increments and of discrete event-based simulations. While simulators with fixed time increments, e.g. agent-based simulators, are well suited for parallelization on GPUs, they concluded that further research is required to increase the performance of discrete event-based simulation frameworks through the usage of GPUs.

According to [6] and [7] GPUs have proven to improve the runtime performance of signal processing algorithms, e.g. for tasks such as bit modulation, fast-fourier transformations or raytracing. The objective of this paper is therefore not to accelerate specific algorithms through GPU-based signal processing, but to improve the overall performance of a discrete event-based simulator that employs signal processing at specific events.

When partitioning the simulation workload and distributing it to multiple processors or cores, correctness of results becomes an issue, as dependencies between simulation events need to be taken into consideration. The *lookahead* metric denotes the time period during which future events can be processed in parallel without violating correctness constraints. Interdependencies between events limiting the lookahead are a result of characteristics of the system being simulated. Kunz et al. [8] proposed a formalism for modeling event interdependencies to facilitate parallel processing by attributing time durations to individual events. Independent events are identified by their overlapping time durations on the simulation timeline. Characteristics of wireless communication scenarios allowing for parallel processing of events are addressed in detail in [9]. Parallelism is exploited based on the radio propagation delay, terrain information and channel characteristics.

III. Accurate Wireless Network Simulation

In [3] we presented an extension to the popular network simulator NS-3 in order to enable an accurate simulation of wireless networks based on IEEE 802.11a, g or p. The extension discards the abstraction of a packet and considers individual bits as well as their corresponding signals. A packet is not only described by its length, data rate and the used transmit power anymore, but by a sequence of up to thousands of complex time samples. As a result, an implementation of the physical layer has to employ and emulate the computationally

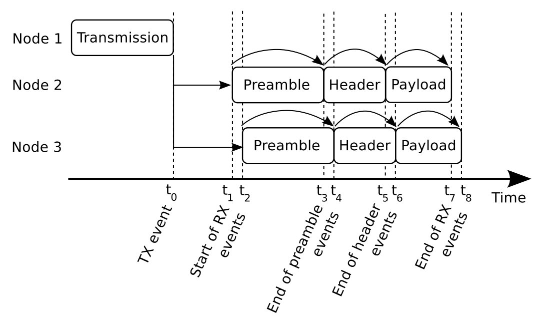

Fig. 2. Sequence of events that is executed in a network simulation in case a node transmits a packet and multiple nodes receive the packet.

expensive signal processing steps that are normally executed in a real transceiver, e.g. convolutional encoding, bit scrambling, interleaving, OFDM modulation, signal detection and channel estimation. Due to the data intensive signal processing the runtime performance of such a simulation is reduced by up to four or five orders of magnitude. To understand now how GPU-based signal processing should be integrated into a discrete event-based network simulator, we will sketch the transmission and reception processes in the following and outline how signal processing is scattered over these events. Due to space restrictions we skip the details of the overall physical layer emulation, refer to [3] for a detailed explanation of the specific signal processing algorithms, and focus on the aspects that are relevant in order to understand our proposed GPU-based architectures.

In order to reflect the behavior of a transceiver correctly and to account for possibly overlapping transmissions, the transmission and reception process has to be divided into several events. As illustrated in Figure 2 a packet transmission will lead to several events at each potential receiver: first, an event that indicates the arrival of the first time sample is scheduled, followed by events that reflect the points in time at which the last time sample of the preamble, the packet header and the payload has been received. At each of those events, a decision will be made whether the reception process of this packet is continued or not. Hence, in case of multiple receivers, there will be a large amount of events that are separated only slightly in time. Since signal processing has to be performed at each single event, it is not obvious whether GPU-based signal processing in a network simulation can generally improve the overall runtime performance. For instance, it is not clear whether and under which conditions the overhead due to data transfers from CPU to GPU and vice versa, as well as due to the delay for context switches between CPU and GPU, is greater or smaller than the performance gains that can be achieved through parallel processing on GPUs.

IV. GPU-based Optimization

A. GPU-based Signal Processing

As a basis for subsequent performance analysis, we ported three computationally expensive signal processing algorithms used for an accurate simulation of wireless network communication to GPUs. For each, we measured the achieved speedup

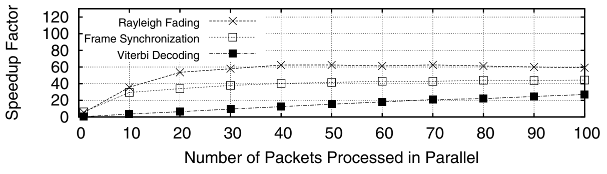

Fig. 3. Speedups achieved by GPU-based parallelization of individual signal processing algorithms

factors when running the algorithms on an ATI Radeon HD 5870 graphics card with 1600 cores in relation to a sequential execution on a single core of an AMD Phenom II X6 1035T CPU. The algorithms operate on packets with a payload of 500 bytes at the lowest data rate – and thus on a large number of complex time samples – each. To analyze the conditions under which significant speedups can be observed, we vary the number of packets processed in parallel between 1 to 100, corresponding to 1 to 100 receivers. Figure 3 illustrates the speedup factors achieved by parallelization. Across all three algorithms it is evident that speedups are marginal when only a small number of packets is processed in parallel. This observation can be attributed to the data transfer and context switch overheads in particular, as well as to the lower clock speed of the GPU cores in general. The benefit of GPU-based signal processing is only significantly increased if the number of packets to be processed in parallel is raised. For instance, when processing 100 packets in parallel, we observed speedup factors of 59.1 for the computation of Rayleigh fading channel effects, 44.3 for frame synchronization and 27.0 for Viterbi decoding. Since recent GPUs and SDKs support double precision floating point arithmetic, the results of the sequential CPU-based execution and the parallelized GPU-based execution differ only in the order of 10^{-15} on average.

From the results, we conclude that substantial speedups can be achieved using GPU-based signal processing. However, we identify a maximization of the amount of input data processed per GPU work cycle as a prerequisite for optimal performance.

B. GPU-based Simulation Architectures

In the following, we present, evaluate and compare four architectures that aim to provide efficient GPU-based signal processing within a discrete event-based network simulator.

The most simple approach of a GPU-based discrete event simulation is depicted in Figure 4a. Events are processed sequentially on the CPU. For time-consuming signal processing tasks, input data is transfered to the graphics card's memory. Once the GPU finishes parallel processing of the task, output data is transfered back to the host computer's main memory. This process is repeated for all signal processing tasks associated with the event. A second and more efficient approach is based on the aggregation and parallel execution of identical tasks that belong to different but independent events, cf. Figure 4b. With this approach, multiple data transfers and context switches are reduced to only one transfer and one context switch. This approach can be optimized even further, if the output of one event serves as the input of the next event, or if subsequent events operate on the same input data. Additional data transfers can then be saved by reusing memory and data that has been transfered to the graphics card at an earlier stage, cf. Figure 4c. Finally, a purely GPU-based simulation architecture can be envisioned (cf. Figure 4d). With this approach, all simulation logic resides on the GPU, reducing data transfers to the beginning and the end of the simulation.

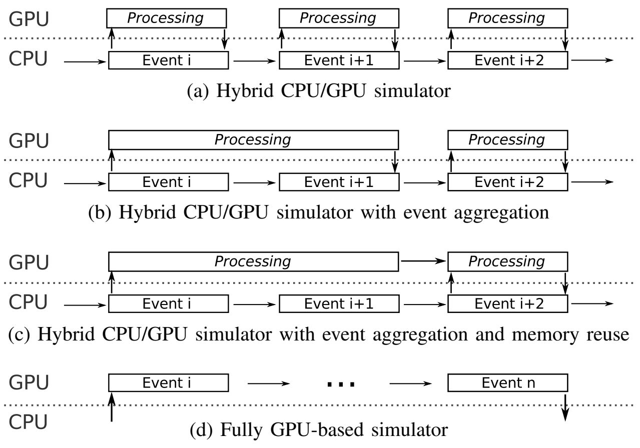
GPU
CPU
Processing
Event i
Event i+1
Event i+2
(a) Hybrid CPU/GPU simulator
(b) Hybrid CPU/GPU simulator with event aggregation
(c) Hybrid CPU/GPU simulator with event aggregation and memory reuse
Event n
(d) Fully GPU-based simulator

Fig. 4. Approaches to GPU-based time-discrete event-based simulation

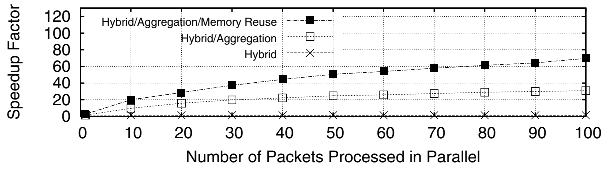

Fig. 5. Comparison of the proposed GPU-based architectures against the sequential processing on CPU.

To evaluate and compare the performance of the four architectures, we developed a discrete event-based simulator from scratch. Based on this simulator, we developed a benchmark that reflects a simple simulation comprised of a chain of three simulation events associated with a single frame transmission and the corresponding receptions. Each event triggers one of the three signal processing algorithms, whereas an algorithm uses the output of the previous algorithm as its input.

Figure 5 depicts the speedups achieved when implementing the GPU-based simulation architectures compared to a sequential processing of events on CPU. As can be seen, a hybrid simulation yields a speedup factor of 1.5 independent of the number of receivers. This demonstrates the impact of overheads involved in frequent crossing of the CPU/GPU boundary. Additional event aggregation yields an overall speedup factor of 30.9 for 100 receivers. Elimination of redundant data transfers by memory reuse further increases the total speedup factor to 69.6 for 100 receivers.

A fully GPU-based simulation is technically not yet feasible, since sufficient support for the synchronization of parallel threads is not provided by current GPU chipsets nor by the SDKs. However, such a synchronization is crucial in order to implement the discrete event-based scheduling and processing logic of the simulator itself.

C. Discussion and Outlook

Based on the obtained results, it looks as if hybrid simulation with event aggregation and memory reuse provides the best runtime performance. In order to answer our primary question – whether the costs for accurate wireless network simulations can be reduced or even eliminated – we applied our findings to the NS-3 physical layer extension sketched in Section III. However, since transparent memory management on GPUs is not yet supported by recent SDKs, we applied only our hybrid approach with event aggregation.

According to our code profiling, the three signal processing algorithms that we ported to the GPU, cf. Section IV-A, contribute to approx. 86 % of the total simulation runtime – considering a network scenario with 100 nodes. Regarding Amdahl's law, the maximum speedup that could theoretically be achieved is therefore 7.1. In practice, however, we only achieved a speedup factor of 4.3, which accounts for approx. 60 % of the theoretical maximum. Can we gain more? The answer is again given by our code profiling, which states that more than 99 % of the total runtime is spent on parallelizable signal processing, which increases the maximum achievable speedup to more than 110. Even if only 60% of this theoretical maximum is achieved, the overall costs in terms of runtime slowdown can be reduced from four down to two orders of magnitude. With an additional reuse of GPU memory, even higher speedups can be expected.

As pointed out above and demonstrated in the previous section, significant speedups for the accurate simulation of wireless networks can be achieved. However, minimization of the overheads depends on features that are not present in existing network simulation frameworks. Therefore, we now discuss the amount of modifications that are needed to establish efficient GPU-based signal processing in such network simulators. The hybrid approach requires only a port of the signal processing algorithms to the GPU. The core logic of the event scheduler remains untouched. As such, our most simple hybrid approach is easy to integrate. To support event aggregation, dependencies between events, e.g. due to model or scenario characteristics, must be considered during simulation in order to maintain correctness of the simulation results. To support a hybrid simulation with memory reuse the GPU's memory has to be managed, either transparently by the SDK or manually by the developer himself. While the first solution is not available yet, the second one is vulnerable to memory leaks and requires a careful handling of memory allocation, pointers and memory swapping by a developer – which is in strong contrast to the objectives of modern simulation frameworks that adhere to "best practices" of software engineering such as modularity of code and usage of design patterns.

In order to enable fully GPU-based simulations, future GPUs and SDKs need to offer a global synchronization between GPU cores. As long as global synchronization is not provided, event schedulers can not be ported to the GPU. Hence, the core simulation logic has to stay on the CPU.

V. Conclusions

In this paper, we studied the potential of GPU-based processing in order to improve the runtime performance of computationally intensive accurate wireless network simulations. We presented, compared and evaluated four simulator architectures that support the integration of GPU-based signal processing into existing or future simulation frameworks. Exemplarily, we parallelized three computationally intensive signal processing algorithms and ported them to GPUs to achieve individual speedup factors of up to 60 compared to sequential execution on a CPU. In a benchmark simulation we studied how efficiently our proposed architectures can integrate the individual optimizations into a discrete event-based simulator and observed speedup factors of up to 70 using the most advanced architecture that is feasible with state-of-the-art GPUs and SDKs. We identified parallel execution of independent events, i.e., event aggregation, and memory reuse as essential mechanisms to minimize the overheads and to increase observable speedups. Furthermore, it is necessary to port as many signal processing algorithms as possible to enable accurate wireless network simulations at only moderate costs.

Acknowledgment: Jens Mittag acknowledges the support of the Ministry of Science, Research and the Arts of Baden-Württemberg (Az: Zu 33-827.377/19,20), the Klaus Tschira Stiftung, the INIT GmbH and the PTV AG for the research group on Traffic Telematics.

References

[1] K. Tan, J. Zhang, J. Fang, H. Liu, Y. Ye, S. Wang, Y. Zhang, H. Wu, W. Wang, and G. M. Voelker, "SORA: High Performance Software Radio Using General Purpose Multi-core Processors," in *Proc. of the 6th USENIX Symposium on Networked Systems Design and Implementation*, 2009, pp. 75–90.

[2] G. Judd and P. Steenkiste, "Repeatable and Realistic Wireless Experimentation Through Physical Emulation," *SIGCOMM Comput. Commun. Rev.*, vol. 34, no. 1, pp. 63–68, 2004.

[3] J. Mittag, S. Papanastasiou, H. Hartenstein, and E. G. Ström, "Enabling Accurate Cross-Layer PHY/MAC/NET Simulation Studies of Vehicular Communication Networks," *Proceedings of the IEEE*, no. 99, pp. 1–16, July 2011.

[4] Z. Xu and R. Bagrodia, "GPU-Accelerated Evaluation Platform for High Fidelity Network Modeling," in *Proc. of the 21st Int'l Workshop on Principles of Advanced and Distributed Simulation*, 2007, pp. 131–140.

[5] K. S. Perumalla, "Discrete-event Execution Alternatives on General Purpose Graphical Processing Units (GPGPUs)," in *Proc. of the 20th Workshop on Principles of Advanced and Distributed Simulation*, 2006, pp. 74–81.

[6] A. Kerr, D. Campbell, and M. Richards, "GPU VSIPL: High-Performance VSIPL Implementation for GPUs," in *High Performance Embedded Computing Workshop*, September 2008.

[7] S. Bai and D. Nicol, "GPU Coprocessing for Wireless Network Simulation," Tech. Rep., 2009.

[8] G. Kunz, O. Landsiedel, J. Gross, S. Götz, F. Naghibi, and K. Wehrle, "Expanding the Event Horizon in Parallelized Network Simulations," in *Proc. of the 18th Annual Meeting of the IEEE/ACM Int'l Symposium on Modeling, Analysis and Simulation of Comput. and Telecomm. Systems*, August 2010, pp. 172–181.

[9] M. Thoppian, S. Venkatesan, H. Wu, R. Prakash, N. Mittal, and J. Anderson, "Improving Performance of Parallel Simulation Kernel for Wireless Network Simulations," in *Proc. of the 2006 IEEE Conference on Military Comm.*, 2006, pp. 2516–2521.