COST Action IC1405
Reversible Computation
Extending Horizons of Computing

Working Group 4 — Report on Case Studies

Grant Period 3 report

Editors:
Ulrik Pagh Schultz and Carla Ferreira

Contributors:
Carla Ferreira, Stefan Kuhn, Ivan Lanese, Markus Schordan,
Ulrik Pagh Schultz, Harun Šiljak, Irek Ulidowski,
Robert Wille, Alwin Zulehner and Miralem Mehic

August 2, 2018

Contents

Preface

This report presents the case studies selected for investigation in the context of the COST Action IC 1405 on reversible computation, collected and edited by Working Group 4. The descriptions of the specific case studies have been provided by the researchers interested in investigating the corresponding case studies. The current status and recent development are described for each of the case studies.

More on the project:

- `http://www.cost.eu/COST_Actions/ict/IC1405`
- `http://www.revcomp.eu/`

1 Overview

1.1 Introduction

As outlined in the Memorandum of Understanding (MoU), the theories, techniques and support software tools delivered by Working Groups 1 to 3 will be validated and their suitability for practical applications will be assessed using case studies. The case studies will be used to provide feedback on the effectiveness of the reversibility-inspired theories, techniques, and solutions, and can point to new applications and research questions.

During the first year of the action five case studies have been selected, with two additional case studies added during the second year. The case studies are developed based on an interest-driven bottom-up approach, matching member research interests with case studies carrying a potential for external collaboration, either with industry or with researchers outside the area of reversible computing.

1.2 Selected case studies

Eight case studies have been selected based on interest from members of the action:

1. Debugging of concurrent and distributed systems
2. Reliability of distributed systems through reversibility
3. Reversibility in chemical reactions
4. Reversible control of robots
5. Simulator for optimistic parallel discrete event simulation
6. Reversibility in massive MIMO
7. Simulation of quantum computations
8. Mapping of quantum circuit to the IBM QX architectures
9. Error reconciliation quantum key distribution protocols

Case studies 1–5 were added at the end of the first year, case study 6 at the end of the second year, while case studies 7–9 were added at the end of the third year.

1.3 Research areas

The research areas covered by the selected case studies are outlined in Table 1. The areas are divided into Reversible Computing (RC) topics and non-RC topics. There is a good coverage of RC topics from all Working Groups.

	RC topics	Non-RC topics
Debugging of concurrent and distributed systems	reversible language, checkpointing, causal-consistent reversibility, causal analysis	debugging, concurrent and distributed systems
Reliability of distributed systems through reversibility	reversible language, automatic recovery, reversible transactions, causal-consistent reversibility	concurrent and distributed systems, transactions, replication
Reversibility in chemical reactions	reversible language, reversible process calculi, out-of-causal order reversibility, local reversibility	modelling of chemical reactions, life sciences modelling
Reversible control of robots	reversible language, inversion, backtracking, reversible debugging, reversible concurrency	robotics, industrial robots, domain-specific languages, modular robots, swarm robots
Simulator for optimistic parallel discrete event simulation	reversible language, out-of-causal order reversibility	simulation, high-performance computing
Reversibility in massive MIMO (Multiple Input Multiple Output)	reversible control, reversible cellular automata, reversible petri nets	MIMO, communication networks, telecommunications, 5G, antenna selection
Simulation of quantum computations	quantum computations, representation of reversible and quantum functionality	decision diagrams, simulation
Mapping of quantum circuit to the IBM QX architectures	reversible and quantum circuits, quantum computers	search algorithms, technology mapping
Error reconciliation quantum key distribution protocols	reversible function representation, local reversibility	coding techniques, communication networks, telecommunications, QKD

Table 1: Research areas covered by the selected case studies

All case studies are based on significant interaction with researchers from areas outside reversible computing. Moveover, the case studies on debugging, robots, simulation, massive MIMO, simulation of quantum computations, mapping of quantum circuit to the IBM QX architectures, and error reconciliation quantum key distribution protocols all have significant potential for industrial collaboration.

2 Case studies

We now present the nine case studies, in each case providing an overall conceptual description, a concrete example, an estimate of the potential impact, and an update (where applicable) on progress in the area.

2.1 Debugging of concurrent and distributed systems

Involved members: Ivan Lanese, Claudio Mezzina, Naoki Nishida, Adrián Palacios, Germán Vidal

2.1.1 Conceptual description

Finding bugs inside software has always been a main activity in the software development life cycle [6], and reversibility can play a role here [23, 37, 30]. In fact, reversibility allows one to go back in the past of the program, looking for the bug that caused a given misbehaviour. Indeed, many debuggers provide features for reversible execution, including GDB [36] and UndoDB [82]. However, current reversible debuggers either do not support concurrency/distribution, or they just fix an interleaving among threads and stick to it. It is not yet clear which is the good way to debug a concurrent and/or distributed system, what is potentially very interesting. Indeed, finding bugs inside concurrent and distributed software is more difficult than inside sequential software [47], since faults may appear or disappear according to the speed of the different processes and of the network communications. The bugs generating these faults, called Heisenbugs, are particularly difficult to find. Note that, by exploring back and forward the same computation one is guaranteed that Eisenbugs that occurred will not disappear (provided that the approach preserves causal dependencies between actions from different threads).

2.1.2 Concrete example

The causal-consistent approach to reversibility [46] allows one to undo any action provided that its consequences, if any, are undone beforehand. This relates reversibility to causality, instead of time, making it more suitable to distributed systems where a unique notion of time may not exist. In debugging, the causal-consistent approach allows one to find bugs by following backward causal dependencies from the misbehaviour to the bug causing it. This is supported by

primitives allowing one to undo a past action, together with all and only its distributed consequences [45]. For instance, if one observes that, e.g., the value of a variable x is 5 instead of the expected 6, one can undo the last assignment to x, reaching a state where the assignment could have been performed. A causal-consistent reversible debugger for the toy language muOz has been presented in [30], and is available at [18].

2.1.3 Potential impact

Debugging is a topic very relevant from the practical point of view, but mostly neglected by academic research. Concerning impact, a 2012 study by the Judge Business School of the University of Cambridge, UK, sponsored by Rogue Wave Software, goes as far as to estimate that the time spent in debugging corresponds to 49.9% of total programming time [6]. According to another recent study [83], the cost of debugging software amounts to $312 billions annually. Hence, any improvement in debugging techniques has potential huge practical impact. Given that systems are becoming more and more complex, and concurrency and distribution is more and more relevant, the relevance of debugging will not decrease in the close future.

There is a significant potential for industrial collaboration with Undo Software (Cambridge, UK), which is a company concentrated in reversible debugging. They are mainly concerned with sequential software, and in case of concurrency their approach is currently to consider a linearization of the execution. One of their main concerns and selling points is the limited overhead of their debugger, both in time and size of the logs. Still, they want to continue improving these features. Their approach is based on taking snapshots of the state at variable rates, according to how likely the current part of the execution will need to be reversed. Most of their work is in low-level optimizations and in providing support for different compilers and operating system configurations. Undo Software currently has a limited interest in concurrency and distribution, with the more relevant open points for them being parallel logging (since sequential logging requires to linearize execution, hence an unacceptable slow down), and debugging of MPI programs (since their customers are interested in this programming library).

2.1.4 Progress

There has been work to make the *rollback* command typical of causal-consistent debugging available in a core subset of the Erlang language [61]. This work has been exploited to build CauDEr [98, 97] a causal-consistent debugger for the above subset of Erlang. The rollback command is also being investigated for the tuple-based distributed coordination language muKlaim [31]. For details we refer to the accompanying COST IC1405 Grant Period 2 report from *Working Group 2 — Software and Systems*.

2.2 Reliability of distributed systems through reversibility

Involved members: Claudio Antares, Carla Ferreira, Adrian Francalanza, Ivan Lanese, Claudio Mezzina, Vasileios Koutavas, Emilio Tuosto

2.2.1 Conceptual description

The emergence of global services, operating over massive distribution of data and computation, has lead to the recent paradigm of cloud computing. In this global context building reliable systems is very hard, but reversibility can be used to achieve reliability in a natural way. If something goes wrong just undo the previous actions and return to a past consistent state. However, in these highly distributed systems there is no notion of global time, so there might not be a clear previous action. Instead, actions are causally ordered. Leveraging on [20], that defined causal-consistent reversibility, this case study will demonstrate the use of reversibility as a recovery mechanism for highly distributed systems. The goal is that a programmer can build a reversible distributed application for free, in the sense s/he just has to program the forward actions and whenever something goes wrong all actions can be reverted automatically. This was already explored in the context of long-running transactions (see [14] for an overview of calculi with support for compensations / backward actions). In long-running transactions, backward actions were defined by the programmer and the relation between forward and backward actions could be quite tenue. In this case study, we would want to explore how backward actions can be determined automatically for concurrent and distributed systems over a shared state.

Another axis to be explored by the case study is the use of replicated databases to achieve service availability and scalability, as it is common practice in global services. In this setting, clients execute operations over the closest database replica and later these operations are propagated to other remote replicas. Ensuring automatic reversibility in this case is even harder as coordination is reduced (on weaker models of consistency coordination is avoided altogether). It should be explored how to ensure reversibility in this new context.

2.2.2 Concrete example

The concrete examples should include two classes of distributed systems. Distributed and concurrent systems that operate over a shared state, and distributed systems that operate over a replicated state. Several well known distributed systems case studies used in the literature can be used: transactional web server for a online store [80], an auction system, and a student registration system. These specific case studies should show the applicability and relevance of reversibility for distributed systems.

2.2.3 Potential impact

Building concurrent and distributed systems is hard and reversible languages can play an important role in helping programmers developing reliable software by providing reversibility by construction. Currently, error handling has to be done explicitly by the programmer, but if reversibility could be achieved by pressing a button it would have an impact on the quality of the programs developed. In fact, reversibility could be used as a way of providing speculative execution in cloud applications.

2.2.4 Progress

Investigation of the *rollback* command typical of causal-consistent debugging [31, 61] is also highly relevant for reliability of distributed systems, since it is key to implementing automated rollback of each node in the system. Complementary to this, automated recovery in distributed systems is being investigated from the point of view of reversibility of multiparty protocols [62]. For details we refer to the accompanying COST IC1405 Grant Period 2 report from *Working Group 2 — Software and Systems*.

In general distributed programs are hard to get right because they are required to be open, scalable, long-running, and tolerant to faults. The recent approaches to distributed software based on (micro-)services where different services are developed independently by disparate teams only serve to exacerbate the problem. Indeed, when services that are meant to be composed together are run in open context, unpredictable behaviours can emerge. This issue makes it necessary to adopt suitable strategies for monitoring the execution and incorporate recovery and adaptation mechanisms, so as to make distributed programs more flexible and robust. The typical approach that is currently adopted is to embed such mechanisms in the program logic, which makes it hard to extract, compare and debug. Cassar et al. propose an approach that employs formal abstractions for specifying failure recovery and adaptation strategies [91]. Although implementation agnostic, these abstractions would be amenable to algorithmic synthesis of code, monitoring and tests. Cassar et al. consider message-passing programs (a la Erlang, Go, or MPI) that are gaining momentum both in academia and industry. Their research agenda consists of (1) the definition of formal behavioural models encompassing failures; (2) the specification of the relevant properties of adaptation and recovery strategy; and (3) the automatic generation of monitoring, recovery, and adaptation logic in target languages of interest. Francalanza et al. give an instance of the framework proposed by Cassar et al. [92] More precisely, their extension imbues the communication behaviour of multi-party protocols with minimal decorations specifying the conditions triggering monitor adaptations. They show how, from these extended global descriptions, one can (i) synthesise actors implementing the normal local behaviour of the system prescribed by the global graph, and also (ii) synthesise monitors that are able to coordinate a distributed rollback when certain conditions (denoting abnormal behaviour) are met.

2.3 Reversibility in chemical reactions

Involved members: Irek Ulidowski, Stefan Kuhn, Gabriel Ciobanu, Bogdan Aman

2.3.1 Conceptual description

Biological reactions, pathways and reactions networks have been studied using various techniques, including process calculi. The initial research focused on reaction rates, by modelling and simulating networks of reactions, in order to analyse or even predict the common paths through the network. Reversibility was not considered there explicitly. Once a more structural approach was taken reversibility started to be taken into account, since it plays a crucial role in many processes, typically by going back to a previous state in the system. For example, if a gene is regulated by a protein binding to and unbinding from it, then reversibility is needed if this to be modelled directly. Such examples can be modelled by backtracking along the path taken to the current state. Beyond backtracking and causal reversibility, there are more general forms of reversibility, specifically the out-of-causal-order reversibility, which make it possible to get to states which cannot be reached by forward reactions alone. Such sequences of forward and reversible reactions are important as they lead to new chemical structures and new reactions, which would not be possible if the out-of-causal reversibility was not used.

Looking at proteins and other macromolecules means working at relatively high levels of abstraction. It also means that such biological entities are often difficult to represent as there is still no complete understanding of their behaviour. Also their interactions with other similar molecules are hard to enumerate as they are sometime not fully understood. This could also mean that some aspects of reversibility are not clearly modelled since we have abstracted too much detail. Therefore, we suggest to consider chemical reactions when studying reversibility in the context of living systems.

Chemical reactions involve various forms of reversibility and offer good case studies for the modelling of reversibility. Specifically, we can find interesting examples of the out-of-causal order reversibility in many chemical reactions, and can study the intermediate steps of such reactions where some bonds are only "helping" to achieve the overall aim of the reaction: are only formed to be broken before the end such reactions. The level of detail we have about such chemical reactions should help with seeing reversibility in action in the ways not obvious at higher levels. As an additional benefit, in many cases a good understanding of the underlying reaction mechanisms exists.

2.3.2 Concrete example

We suggest to study the following three types of reactions:

- The hydration of formaldehyde in water: This is a basic textbook reaction, but there are different paths through the reaction, some quite rare, and the reaction is reversible. Clearly backtracking is not sufficient to model

this, since it is experimentally possible to go forward via one path and undo reactions via a different path.

- The Monsanto process: This is a typical catalytic cycle involving a metal catalyst. We have two input molecules and want to produce another molecule by combining them. They cannot interact directly because too much energy would be required. By introducing the metal catalyst this can be overcome the catalyst breaks the input molecules and enables their recombination. The catalyst is restored to its original form after the process making it a cycle. Clearly, there must be undoing of bonds created by the catalyst. If fact, some bonds are undone via the out-of-causal order computation.

- The glucose-fructose isomerization: This is a reaction forming part of the Krebs cycle, an important pathway in the metabolism of organisms. It is an enzyme-assisted catalytic process, where an enzyme interacts with the glucose to produce fructose. Similarly to the Monsanto process the protein emerges unchanged. The mechanism, however, involving both causal and out-of-causal order reversibility, is different.

These examples display various types of reversibility, including the out-of-causal order reversibility. All examples have been studied extensively and are well understood at a very detailed level. There are also aspects of reactions which may be included at a later stage, for example the energy levels of the various steps of reactions.

The aim is to model these reactions using different existing techniques. We currently envisage the use of the kappa-calculus [21], P systems [13, 68] and reversible process calculi [19, 45, 67], including the recently developed [39]. The case study should give a clearer view of what the individual calculi can do for each of the examples and clarify their strengths and weaknesses.

2.3.3 Potential impact

The work can help to identify work which needs to be done to get a wider range of reversible processes represented in computing. Furthermore it can help with building knowledge about chemical reactions. Regarding the potential for external/industrial collaboration, there is interest in reaction simulation in the chemoinformatics area. A collaboration could be a long-term goal.

2.3.4 Progress

Recent progress in the area of reversibility in chemical reactions was presented at the COST IC1405 meeting in Belgrade. Based on previous work a process calculus called Calculus for Covalent Bonding (CCB) has been developed which can model many features found in chemical reactions [40]. The calculus has formal semantics given by SOS rules that extended with appropriate predicates. Simulation software was developed based on CCB and used to successfully model several examples of biochemical reactions of increasing complexity.

In order to explore the calculus further we have also modelled a high-level example, Base Excision Repair (BER) of DNA using CCB. This demonstrates that the principles encoded in the calculus are useful beyond the domain where the initial inspiration came from. The software was used to check the modelling. We have identified spatial configurations as an area for further work in this and other examples.

Natural computing is a complex field of research dealing with models and computational techniques inspired by nature that helps us in understanding the biochemical world in terms of information processing. The articles [1] and [2] investigate the reversibility of biochemical reactions in two important theories of natural computing inspired by the functioning of living cells, namely in membrane computing [13, 68] and in reaction systems [26]. Membrane computing deals with multisets of symbols processed in the compartments of a membrane structure according to some multiset rewriting rules; some of the symbols (presented with their multiplicity within the regions delimited by membranes) evolve in parallel according to the rules associated with their membranes, while the others remain unchanged and can be used in the subsequent steps. The situation is different in reaction systems; these systems represent a qualitative model, and so they deal with sets rather than multisets. Two major assumptions distinguish the reaction systems from the membrane systems: (i) threshold assumption claiming that if a resource is present in the system, then it is present in a "sufficient amount" such that several reactions needing such a resource are not in conflict (this means that reaction systems have actually an infinite multiplicity for their resources); (ii) no permanency assumption claiming that an entity disappears from the current state unless it is produced by one of the reactions enabled in that state.

The important innovations of these articles are given by adding the reverse rules to the initial set of rules, as well as by adding an external control specified by using a special symbol informing the system that a rollback is needed. Their contribution is done by several theoretical results relating the evolutions of these systems to their reversible extensions.

2.4 Reversible Control of Robots

Involved members: Ulrik Pagh Schultz, Gabriel Ciobanu

2.4.1 Conceptual description

Robots normally have one or more degrees of freedom controlled by a computational process; using reversible computing to control the robot potentially gives rise to new reverse behaviours. For example, major industrial robot manufacturers such as ABB and KUKA offer limited forms of ad-hoc reverse execution for interactive programming and debugging, but due to limitations in the underlying execution models, their programming models are incapable of reversing complex actions such as steps of an assembly process [49, 51]. We attribute the ad-hoc limitations to the lack of an underlying reversible model. The first in-

vestigation of fully reversible robot behaviours was for self-reconfigurable robots [72]. To better understand the underlying relation between reversible computation and physical reversibility, we will investigate reversible control of industrial robots and distributed robots.

Programming industrial robots is challenging due to the difficulty of precisely specifying general yet robust operations. As the complexity of these operations increases, so does the likelihood of errors. Certain classes of errors during industrial robot operations can however be addressed using reverse execution, allowing the robot to temporarily back out of an erroneous situation, after which the operation can be automatically retried. Specifically, this approach has been shown to be useful for automatic error recovery for small-sized batch production of assembly operations [76]. Moreover, reversibility can in this case be used to automatically derive a disassembly sequence from a given assembly sequence, or vice versa. These results were demonstrated using an initial design and implementation of a reversible domain-specific languages (DSL) for specifying such assembly sequences [76, 49]. The area however remains largely unexplored, both from a theoretical and practical point of view. There is for example a large design space for different programming language approaches, both in terms of the generality of the language and the means by which reversibility is achieved. At a more fundamental level, the notion of reversible control of a reversible physical system remains largely unexplored. From a practical point of view, only the specific case of assembly operations has been investigated, and only using a specific set of industrial use cases. There has been no attempt at integration into an existing robotics platform, although we observe that many existing platforms offer limited notions of reversibility for using during programming and debugging.

Distributed robotic systems can make use of reverse execution as a means to reversing physical actions or correcting errors. As a concrete example, previous work has studied self-reconfigurable, modular robots which are distributed mechatronic devices that can autonomously change their physical shape. Self-reconfiguration from one shape to another is typically achieved through a specific sequence of actuation operations distributed across the modules of the robot. Automatically reversing the sequence of operations brings the robot back to its initial shape, which has been experimentally demonstrated using the DynaRole reversible language [72]. DynaRole however only allows simple sequences of operations to be reversed, which is suitable for reversing self-reconfiguration sequences but lacks the generality needed to implement more complex behaviors. Initial ideas on generalizing the DynaRole language to support a wider range of scenarios, while retaining the possibility of reversing distributed sequences, have been proposed [77]. Reversibility is used here as a practical feature, reducing the programming task of the programmer, and allowing error recovery by backing out of an error state using reverse execution. Currently there however is no underlying model allowing a distributed robotic systems to use reversibility for backing out of distributed operations that have become blocked, e.g., due to unexpected features of the environment, to under conflicting operations, or recover from partial hardware failures.

2.4.2 Concrete example

An appropriately robust and useful reversible DSL for control of industrial robots could be integrated into an existing robotics platform, and could be useful both for working interactively with the robot (programming and debugging) but also for error recovery during specific kinds of operations. Note that many robotic systems already integrate limited forms of reverse execution for interactive debugging [51], better understanding and improving these approaches to interactive debugging is probably interesting.

Using reversible computing to allow a subset of a distributed robotic system to optimistically proceed with an operation, but then backtrack and undo the operation if the system globally decides not to perform this operation. This is interesting not only for modular robots but also for swarm robotic system e.g. drones. A single mobile robot navigating an environment can be modeled as a 2D Turing machine [81], presumably a useful model for multi-robot systems navigating an environment could be based on a similar model.

2.4.3 Potential impact

From a scientific point of view, the design of reversible languages for controlling reversible processes remains an open question. Moreover, exploring reversible control of physical processes may help to better understand the principles of reversible computing. From a society point of view, industrial robots are key to maintaining production in Europe, and reversible computation has the potential to increase robustness for specific kinds of operations such as assembly, and moreover facilitate the programming of such operations.

There is a significant potential for external/industrial collaboration for industrial robots: Universal Robots ApS is a potential industrial collaborator, as are potentially also other robotics companies or perhaps companies performing large numbers of error-prone assembly operations. For modular robots, we however note that swarm and modular robots are mostly at the level of basic research in robotics, with the obvious exception of drones (but here reversibility seems less useful).

2.4.4 Progress

Significant progress in the area of reversibility for industrial robots was presented at COST IC-1405 meeting in Belgrade [48]. Key developments include an improved understanding of the interaction between reversible computing and real-world systems that only are partially reversible, as well a substantial experimental evaluation of the use of reversible languages to control industrial robots performing assembly and disassembly in the context of small-batch production. Overall this work experimentally demonstrates the use of reversible computing to improve system reliability.

2.5 Simulator for optimistic parallel discrete event simulation

Involved members: Markus Schordan

2.5.1 Conceptual description

Discrete event simulation (DES) is a simulation paradigm suitable for systems whose states are modeled as changing discontinuously and irregularly at discrete moments of simulation time. DES is event-driven in that the times at which state changes occur are calculated dynamically rather than statically as in time-stepped simulations. Example applications include simulations of digital communication networks, vehicular traffic flow, markets and economies, epidemiological models, logistical models, ecological and population models, tactical and strategic military models, and many others. Most systems whose behavior is not describable by continuous equations and that are not suitable for simple time-stepped models are candidates for DES.

Reverse computation has become a central notion in parallel discrete event simulation (PDES) over the last decade [15, 65]. It is not just a theoretical line of research, but an immensely practical one necessary to achieve high performance for large parallel discrete event simulations. In fact, the most highly parallel and fastest discrete event simulation benchmarks ever executed have made essential use of reverse computation [5].

The fundamental issue that makes PDES so complex is the synchronization problem. Every logical process (LP) must execute all events from its own event queue in strictly non-decreasing timestamp order despite the fact that it does not generally know which LPs might be sending it events, or how many, or what timestamps they may carry. Furthermore there is no guarantee that event messages will arrive at an LP in increasing timestamp order. Optimistic synchronization is that all event executions are speculative or provisional, and are always subject to rollback if the simulation gets into local synchronization trouble. Most of the time that does not happen and the simulation proceeds forward in parallel, but occasionally a causality violation occurs that has to be corrected by rollback (achieved by reversible computation) and then the simulation continues forward again. Therefore, reversible computation is paramount for achieving high-performance in optimistic parallel discrete event simulations on supercomputers. Various approaches to reversibility have been developed over the years and have they been successfully applied in supporting optimistic PDES, with recent contributions in incremental state-saving [79, 75], reverse code generation [66, 43], utilization of transactional memory [74], and hybrid combinations of these approaches [69, 16]. A recent publication addresses also instrumentation of binary codes [17].

2.5.2 Concrete example

We suggest to use an existing simulator as basis for optimistic PDES which allows to plug-in reverse code for implementing the rollback as required for

optimistic PDES. A well known simulator is ROSS (Rensselaer's Optimistic Simulation System), a general purpose discrete event simulator developed at RPI by C. Carothers et. al. [34].

ROSS helped to demonstrate that Time Warp simulators can be run efficiently both in terms of speed and memory usage relative to a high-performance sequential simulator and was used to achieve the world-record in 2013 for the most highly parallel and fastest discrete event simulation benchmarks ever executed [5]. In this strong scaling study of the ROSS simulator running Time Warp with reverse computation, had 251 million PHOLD logical processes and was executed in several configurations with up to a peak of 7.86 million MPI tasks running on 1,966,080 cores. At the largest scale it processed 33 trillion events in 65 seconds, yielding a sustained speed of 504 billion events/second using 120 racks of Sequoia. This is by far the highest event rate reported by any parallel discrete event simulation to date, whether running PHOLD or any other benchmark.

In order to use Time Warp in a ROSS model, a reverse event function must be provided. For a given event message it is responsible for undoing the state changes that the forward event function incurred for the same message. ROSS is open-source and also comes with a number of models (implemented in C/C++) with efficient hand-written reverse functions. These can be used as a basis for comparing different automated approaches to reversibility and perform comparative evaluations with the existing reversible ROSS implementations.

2.5.3 Potential impact

It is important to investigate how different approaches to reversibility can attack the problem of making the execution of a model's event function reversible, because any overhead introduced in the forward execution or reverse computation (dependent on the approach to reversibility) impacts the performance of the simulation. In particular, the combination of approaches is interesting for large-scale simulation models, as they are usually a combination of components with different computational properties and challenges for reversibility. If successful, the impact would be that this can bring also other approaches (from the COST project) to be used — or at least to be evaluated to be suitable — to HPC supercomputers, and may also help to improve the performance of simulations. In particular, also reversible languages such as Janus [88] that can be translated to C are candidates for integration with other approaches, and can be directly evaluated and compared to existing models utilizing reverse code generation. There is potential for industrial collaboration with Lawrence Livermore National Laboratory.

2.5.4 Progress

The Backstroke compiler for generating reversible programs from C++ was released to the public in March 2017 (version 2.1.0). This was the first public release of Backstroke V2 using incremental state saving. Moreover the combi-

nation of incremental state saving (i.e., using Backstroke) with Janus-generated reversible C code was investigated during an STSM by Michael Kirkedal visiting Markus Schordan at Lawrence Livermore National Laboratory.

The work on Backstroke has progressed by applying it successfully also to C++ Standard types and the Kinetic Monte-Carlo model that was also presented in [79]. In this crystal grain simulation, a piece of solid is modelled as a grid of unit elements. Each unit element represents a microscopic piece of material, big enough to be able to exhibit a well defined crystal orientation, but much smaller than typical grain sizes. These unit elements are commonly called spins, since the nature of grain evolution resembles evolution of magnetic domains. In this new experiment the model was run at a much bigger scale with 1536×1536 spins in 256×256 logical processes (LPs). The model's implementation was also adapted by removing all hand-written code that was previously used to preserve destroyed information for the hand-written reverse code. In this new experiment the model is implemented using C++ Standard containers and algorithms and user-defined types. After the tranformation by Backstroke the model was run for 2 time units, or a total of 47633718 events on LLNL's IBM BlueGene/Q supercomputer with 16 cores per node, using up to 8192 cores.

2.6 Reversibility in Massive MIMO

Involved members: Harun Šiljak

2.6.1 Conceptual description

Massive MIMO (Multiple Input Multiple Output) systems have been introduced less than a decade ago as a theoretical concept in telecommunications [52], but since then it has entered the practical realm as well ad established itself as one of primary technologies for the coming 5G framework [7]. In brief, it is a concept expanding on classical MIMO by using tens and hundreds of antennas which may be distributed in the environment (distributed massive MIMO) or placed on a single panel (collocated massive MIMO).

Since massive MIMO calls for solving several control and optimisation problems in its application, we aim to introduce reversible computing in the massive MIMO context through the concept of reversible control. In addition to that, since modelling of massive MIMO and networks created by it is of significant interest, reversibility might be of use there as well. In particular, reversible cellular automata [59] and reversible Petri nets [8] will be investigated as a way of massive MIMO modelling, because of their flexibility and novelty. In case of massive MIMO, ability to keep track of reversible cellular automaton's history, undo actions and run it backwards can prove useful in a dynamical environment (periodic changes in environment or mobile station movements benefiting from running antenna selection backwards, algorithms defined on a higher level of abstraction benefiting from undo in case of cost function decrease).

Another natural question to pose is the relationship between reversibility and transmitter-receiver duality [84] and the question of its applicability in practice.

2.6.2 Concrete example

There have been some recent efforts in solving the problem of antenna selection in massive MIMO [33, 28] using cellular automata (work in progress by Harun Šiljak, proposer of this case study). Due to the dynamical nature of the environment and movement of user equipment in it, it would only be natural to allow these selection algorithms to make a step backward or a step forward from its current state to suit the needs of the user equipment in a new configuration which may have occured earlier as well. This reversible control problem could be, for instance, solved by moving from classical cellular automaton modelling the antenna selection process to a reversible one.

2.6.3 Potential impact

There are several different potential impacts of this study. First of all, placing massive MIMO (and communication networks in general [57]) and reversibility in the context of complex systems science will help us investigate relationship between emergent behaviour and reversibility, a question which has drawn some interest in the past as well [42, 35]. Second, the interest for reversibility in telecommunications is mostly spurred by the need for energy efficiency [50, 32] offered by reversible circuits. This study will offer a fresh perspective on reversibility in this context. Last, a third benefit is the possible direct field application of reversible control algorithms for massive MIMO prototypes enhancing their performance.

2.6.4 Progress

In the COST IC-1405 meeting in Larnaca two applications of reversible computing were presented in brief, both focusing on the underwater acoustic communications. First is the application of reversible cellular automata for modelling, and the second is a reversible hardware implementation of wave time reversal. The choice of underwater use case is motivated by the inherent reversibility of the water environment and the possibility of sound wave propagation modelling using cellular automata. The hardware implementation of time reversal will remain in focus of the study and will be developed in more detail in the future. In terms of radio-wave Massive MIMO, this implementation will be evaluated for its applicability for conjugate beamforming.

2.7 Simulation of Quantum Computations

Involved members: Alwin Zulehner, Robert Wille

2.7.1 Conceptual description

Quantum computation is a promising emerging technology which is inherently reversible and, compared to conventional computation, allows for substantial speed-ups e.g. for integer factorization or database search. However, since physical realizations of quantum computers are in their infancy, a significant amount of research in this domain still relies on simulations of quantum computations on conventional machines. This causes a significant complexity which current state-of-the-art simulators (e.g. [93, 110, 107, 94, 96, 102, 108]) try to tackle with a rather straight-forward array-based representation and by applying massive hardware power. There also exist solutions based on decision diagrams (i.e. graph-based approaches [109, 104, 95]) that try to tackle the exponential complexity by exploiting redundancies in quantum states and operations. However, these existing approaches do not fully exploit redundancies that are actually present.

In this case study, we investigate how corresponding quantum states and quantum operations can be represented even more compactly, and, eventually, simulated in a more efficient fashion. A first realization in this direction (which is publicly available at `http://www.jku.at/iic/eda/quantum_simulation`) has shown to be capable of simulating quantum computations for more qubits than before, and in significantly less run-time (several magnitudes faster compared to previously proposed simulators) [115].

2.7.2 Concrete example

In this case study we investigate how corresponding quantum states and quantum operations can be represented compactly and simulated efficiently. In fact, the general idea is motivated by the decomposition scheme of Quantum Multiple-Valued Decision Diagrams (QMDDs [101]) – a type of decision diagram, that is frequently used in the quantum and purely Boolean reversible domain. In fact, the natural decomposition of QMDDs, enriched with dedicated manipulation algorithms required for simulating quantum computations leads to an efficient simulation approach whenever redundancies can be exploited.

2.7.3 Potential impact

By reducing the effort for simulating certain quantum computations, our simulation approach allows for research on quantum algorithms, even though no sophisticated physical realizations are available yet. Moreover, the proposed approach allows for verifying future quantum computers in less runtime. Moreover, the application area for the underlying representation of quantum computations and state vectors shall be broadened to other areas of the quantum domain in order to allow for computations that are intractable using the existing methods.

2.7.4 Progress

Using a first implementation of the proposed design methodology [115], certain quantum computations can be simulated for more qubits than before, and in significantly less run-time (several magnitudes faster compared to previously proposed simulators). A project proposal for research on further improving the proposed simulation approach and broadening the application area of the underlying data structures and manipulation algorithms has been funded by Google's Research Award Program.

2.8 Mapping of Quantum Circuit to the IBM QX Architectures

Involved members: Alwin Zulehner, Alexandru Paler, Robert Wille

2.8.1 Conceptual description

In the past years, quantum computers more and more have evolved from an academic idea to an upcoming reality. IBM's project *IBM Q* can be seen as evidence of this progress. Launched in March 2017 with the goal to provide access to quantum computers for a broad audience, this allowed users to conduct quantum experiments on a 5-qubit quantum computer and, since June 2017, also on a 16-qubit quantum computer (called *IBM QX2* and *IBM QX3*, respectively). Revised versions of these 5-qubit and 16-qubit quantum computers (named *IBM QX4* and *IBM QX5*, respectively) are available since September 2017. In order to use these, the desired quantum functionality (e.g. provided in terms of a quantum circuit) has to be properly mapped so that the underlying physical constraints are satisfied. This constitutes a complex task. One issue is that the desired functionality (usually described by higher level components) has to be decomposed into elementary operations supported by the *IBM QX* architectures. Furthermore, there exist physical limitations, namely that certain quantum operations can only be applied to selected physical qubits of the *IBM QX* architectures. Consequently, the logical qubits of a quantum circuit have to be mapped to the physical qubits of the quantum computer such that all operations can be conducted.

2.8.2 Concrete example

Since it is usually not possible to determine a mapping such that all constraints are satisfied throughout the whole circuit, this mapping may change over time. To this end, additional gates, e.g. realizing SWAP operations, are inserted in order to "move" the logical qubits to other physical ones. They affect the reliability of the circuit (each further gate increases the potential for errors during the quantum computation) as well as the execution time of the quantum algorithm. Hence, their number should be kept as small as possible.

While there exist several methods for efficiently mapping higher level components to elementary operations (see [90, 100, 99]), there is hardly any work on

how to efficiently satisfy the additional constraints for these new and real architectures. Although there are similarities with recent work on nearest neighbor optimization of quantum circuits as proposed in [112, 103, 111, 105, 106, 113], they are not applicable since simplistic architectures with 1-dimensional or 2-dimensional layouts are assumed in that work which have significantly less restrictions. Even IBM's own solution, which is provided by means of the Python SDK *QISKit* [89] fails in many cases since the random search employed there does not cope with the underlying complexity and cannot generate a result in acceptable time.

2.8.3 Potential impact

With growing sizes of physical realizations of quantum computers, automated methods for mapping (or compiling) quantum circuits (quantum algorithms) to physical devices become essential. While the first step, which includes decomposing the quantum functionality to elementary operations has already been intensely studied by previous work, there have hardly been proposed any mapping algorithms for these upcoming devices. However, only a combination of these two types of algorithms is required in order to compile real quantum algorithms for future quantum computers in an efficient fashion.

2.8.4 Progress

In [114], a basic methodology for solving the mapping problem has been proposed which addresses the mapping problem, in a generic way, which can easily be configured for similar future architectures, and is fully integrated into IBM's SDK. Experimental evaluations in [114] show that the proposed approach clearly outperforms IBM's own mapping solution. In fact, for many quantum circuits, the proposed approach determines a mapping to the IBM architecture within minutes, while IBM's solution suffers from long runtimes and runs into a timeout of 1 hour in several cases. As an additional benefit, the proposed approach yields mapped circuits with smaller costs (i.e. fewer additional gates are required).

2.9 Error reconciliation quantum key distribution protocols

Involved members: Miralem Mehic

2.9.1 Conceptual description

Quantum Key Distribution (QKD), based on the laws of physics rather than the computational complexity of mathematical problems, provides an information-theoretically secure (ITS) way of establishing symmetrical binary keys between two geographically distant users. The keys are secure from eavesdropping during transmission and QKD ensures that any third party's knowledge of the key is reduced to minimum [4, 70]. QKD employs two distinct channels between

communicating parties: the quantum channel, which is used for transmission of quantum key material encoded in certain photon properties such as polarization or phase, and the public channel, which is used for verification of exchanged key material. The combination of these two channels forms a QKD communication link, over which QKD allows two remote users to exchange specific type of data, for example, secret keys.

The key material establishment process is performed using a QKD protocol to provide the key to a remote user in a safe manner. Generally speaking, QKD protocols can be roughly distinguished into three broad categories: discrete-variable protocols (BB84, B92, E91, SARG04), continuous-variable (CV-QKD) protocols and distributed-phase-reference coding (COW, DPS) [78, 3]. It is important to note that QKD protocols consist of nearly identical steps at a high level, but differ, among others, in the way the quantum particles or photons are prepared and transmitted over the quantum channel [38, 53, 54]. Although there are differences in implementations, almost every QKD post-processing includes following steps: extraction of the raw key (sifting), quantum bit error rate (QBER) estimation, error reconciliation, privacy amplification and authentication [78].

QKD starts with the transmission of photons over a quantum channel. The obtained raw key material is pushed to sifting module to the first analysis and preliminary processing. The sifted key may contain errors due to background photons, detector noise, polarization imperfections or eavesdropping influence. To correct or delete those errors, the sifted key is further forwarded to QBER estimator and key reconciliation module [22, 29]. To reduce potential information leakage in a quantum transmission phase and to strengthen key's privacy, some bits of the key are discarded in privacy amplification stage. Finally, the processed key is authenticated using a hash algorithm to verify its symmetry on both sides and it is stored in key material storages. The key is subsequently used to secure user's data communications [38].

As shown in traffic analysis experiments, error key reconciliation represents a highly demanding part of the whole process [55]. Since the final output from the whole process is a symmetric cryptographic key, all the errors should be fixed at this stage. The cascade protocol is a well-known key reconciliation protocol which can operate at improved efficiencies and also at high Mbit/s throughput rates [58]. Cascade begins with the random permutation of the key with the objective to evenly disperse errors throughout the key. The permuted key is divided into equal blocks, and cascade continues to run iteratively in the given number of iterations. After each iteration, permutations are performed again, and the block size is doubled. For each block, cascade will compare the results of the parity-check test and perform a binary search to find and correct errors. The process is recursive, and instead of going through all the iterations continuously, cascade investigates and corrects errors in pairs of iterations (hence the name).

2.9.2 Concrete example

When communication parties are sure that the key distributed via the quantum channel has a low error rate by comparing the estimated QBER with the well known QBER of the channel, they need to find and correct or delete all errors in the rest of the key. This phase is known to be highly interactive and time-consuming since the discussion about the location of errors in the key is performed through the public channel. The cascade protocol [12] is the most widely used reconciliation protocol due to its simplicity and efficiency. It is run iteratively in the given number of iterations where random permutation of the key is performed with the objective to evenly disperse errors throughout the key. Next, the permuted key is divided into equal blocks of k_i bits, and after each iteration, permutations are performed again, and the block size is doubled: $k_i = 2 \cdot k_{i-1}$. For each block, communication parties exchange the results of the parity test and perform a binary search to find and correct errors. Instead of going through all the iterations continuously, the cascade protocol investigates errors in pairs of iterations. The process is recursive. This means that no bits are discarded during the first iteration. It also means that for any error corrected in the second iteration there must be at least one matching error contained in the same block in the previous iteration since neither error was found or corrected in that iteration. For this reason, for each correction made in any iteration after the first one, a binary search is rerun on the block containing the bit corrected in all previous iterations, to identify any potential matching errors. For any new error detected, it follows that another error in a previous iteration was masked. Thus the process is repeated so that the error detection and correction process cascades through all previous iterations.

The length of the initial block k_1 is a critical parameter and should depend on the estimated error rate. An empirical result in [12] indicates that the optimal value of k_1 is 0.73/p, where p is the estimated QBER. The cascade protocol is modified in [73], with the aim of reaching the theoretical limit for protocol efficiency. From these results, it is clear that four iterations are sufficient for a successful key reconciliation, as suggested in [12]. However, since the initial block length depends on the estimated error rate, it is necessary to perform all iterations. The number of iterations i is increased to the value for which the length of block k_i can be used to split the raw key into two parts ($k_i < \frac{n}{2}$). Now, let us go back to the parity check results. If the parity of a block disagrees between communication parties, they perform a binary search on that block with the aim of identifying the single bit error. The binary search consists of dividing the block in half and comparing the parity check results for the divided block until the error is located. This means a maximum of $1 + \lceil \log_2 k_i \rceil$ parity bits are exchanged for each block with an error bit, since $1 + \lceil \log_2 k_i \rceil$ is the maximum number of times block k_i can be divided, and one parity bit is exchanged for blocks without errors. From the results presented in [73], it is evident the majority of errors are corrected in the first two iterations.

2.9.3 Potential impact

The cascade protocol is simple and probably the most widely used in QKD implementations. Despite well-known limitations (highly interactive which makes it very sensitive to network latency), the cascade is one of widely used reconciliation protocol in practice due to its relative simplicity and efficiency for relatively low-rate discrete-variables QKD setups [24, 25, 38]. In contrast to cascade, modern reconciliation protocols based on forward error correction methods, such as LDPC or polar codes, are non-interactive. They do, however, require more computation than cascade due to their iterative nature and complex calculations involved with their decoding algorithms [25, 60, 78].

Considering our previous measurements that showed significant interaction and the impact of the cascade protocol on the entire QKD post-processing process [55], it is important to investigate how different approaches to reversibility can reduce the iterations of the cascade protocol. In particular, we are interested in ways of uniformly positioning errors in the key for simpler and more efficient operation using simulation and emulation techniques [56].

References

[1] Bogdan Aman, Gabriel Ciobanu. Reversibility in Parallel Rewriting Systems. *Journal of Universal Computer Science*, 23(7): 692-703, 2017.

[2] Bogdan Aman, Gabriel Ciobanu. Controlled Reversibility in Reaction Systems. *Lecture Notes in Computer Science*, 10725: 40-53, 2018.

[3] Van Assche, Gilles. Quantum cryptography and secret-key distillation. Cambridge University Press, 2006.

[4] Bennet, Charles H. "Quantum cryptography: Public key distribution and coin tossing." Proc. of IEEE Int. Conf. on Comp., Syst. and Signal Proc., Bangalore, India, Dec. 10-12, 1984. 1984.

[5] Barnes, Jr., P.D., Carothers, C.D., Jefferson, D.R., LaPre, J.M.: Warp speed: Executing time warp on 1,966,080 cores. In: Proceedings of the 2013 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation. SIGSIM-PADS '13, New York, NY, USA, ACM (2013) 327-336.

[6] T. Britton, L. Jeng, G. Carver, P. Cheak, and T. Katzenellenbogen. Reversible debugging software - quantify the time and cost saved using reversible debuggers. `http://www.roguewave.com`, 2012.

[7] F. Boccardi, R. W. Heath, A. Lozano, T. L. Marzetta, and P. Popovski. Five disruptive technology directions for 5g. *IEEE Communications Magazine*, 52(2):74–80, February 2014.

[8] Kamila Barylska, Maciej Koutny, Łukasz Mikulski, and Marcin Piatkowski. Reversible Computation vs. Reversibility in Petri Nets. In *Reversible Computation*, pages 105–118. Springer, Cham, July 2016.

[9] L. Benini, A. Macii, M. Poncino, and R. Scarsi. Architectures and synthesis algorithms for power-efficient businterfaces. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 19(9):969–980, 2000.

[10] L. Benini, G. D. Micheli, E. Macii, M. Poncino, and S. Quer. Power optimization of core-based systems by address bus encoding. *Trans. VLSI Syst.*, 6(4):554–562, 1998.

[11] L. Benini, G. D. Micheli, D. Sciuto, E. Macii, and C. Silvano. Address bus encoding techniques for system-level power optimization. In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 861–866, 1998.

[12] Brassard, Gilles, et al. "Secret-key reconciliation by public discussion." Workshop on the Theory and Application of of Cryptographic Techniques. Springer, Berlin, Heidelberg, 1993.

[13] G. Ciobanu, R. Desai and A. Kumar. Membrane systems and distributed computing. In Membrane Computing. International Workshop, WMC-CdeA 2002 Curtea de Arges 2002, pages 187-202. Springer, 2003.

[14] Christian Colombo and Gordon J. Pace. 2013. Recovery within long-running transactions. ACM Comput. Surv. 45, 3, pages 28:1 28:35

[15] Carothers, C.D., Perumalla, K.S., Fujimoto, R.M.: Efficient optimistic parallel simulations using reverse computation. ACM Trans. Model. Comput. Simul. 9(3) (July 1999) 224-253.

[16] D. Cingolani, A. Pellegrini, and F. Quaglia. Transparently mixing undo logs and software reversibility for state recovery in optimistic pdes. In Proceedings of the 3rd ACM SIGSIM Conference on Principles of Advanced Discrete Simulation, SIGSIM PADS '15, pages 211-222, New York, NY, USA, ACM, 2015.

[17] Dealing with Reversibility of Shared Libraries in PDES Davide Cingolani, Alessandro Pellegrini, Markus Schordan, Francesco Quaglia, David R. Jefferson. Proceedings of the 2017 annual ACM Conference on SIGSIM Principles of Advanced Discrete Simulation, SIGSIM-PADS 2017, NIU, Singapore, May 24-26, 2017, ACM. To appear.

[18] CaReDeb `http://www.cs.unibo.it/caredeb`

[19] V. Danos and J. Krivine. Formal molecular biology done in CCS-R. In Proceedings of the 1st Workshop on Concurrent Models in Molecular Biology BioConcur 2003, volume 180 of ENTCS, pages 3149, 2007.

[20] V. Danos and J. Krivine. Reversible communicating systems. In CONCUR04, volume 3170 of LNCS. Springer, 2004.

[21] V. Danos and C. Laneve. Formal molecular biology. Theoretical Computer Science, 325(1):69110, 2004.

[22] Dusek, Miloslav, et al. "Quantum cryptography." Progress in Optics 49 (2006): 381-454.

[23] J. Engblom, A review of reverse debugging, in: System, Software, SoC and Silicon Debug, IEEE, 2012, pp. 1-6.

[24] Elliott, Chip, et al. "Current status of the DARPA quantum network." Quantum Information and Computation III. Vol. 5815. International Society for Optics and Photonics, 2005.

[25] Elkouss, David, et al. "Efficient reconciliation protocol for discrete-variable quantum key distribution." Information Theory, 2009. ISIT 2009. IEEE International Symposium on. IEEE, 2009.

[26] A. Ehrenfeucht and G. Rozenberg. Reaction Systems. Fundamenta Informaticae 75(1), 263280 (2007).

[27] A. Garca-Ortiz, D. Gregorek, and C. Osewold. Optimization of interconnect architectures through coding: A review. In *Electronics, Communications and Photonics Conference*, pages 1–6, April 2011.

[28] X. Gao, O. Edfors, F. Tufvesson, and E. G. Larsson. Massive MIMO in Real Propagation Environments: Do All Antennas Contribute Equally? *IEEE Transactions on Communications*, 63(11):3917–3928, November 2015.

[29] Gisin, Nicolas, et al. "Quantum cryptography." Reviews of modern physics 74.1 (2002): 145.

[30] E. Giachino, I. Lanese, C. A. Mezzina: Causal-Consistent Reversible Debugging. FASE 2014: 370-384

[31] E. Giachino, I. Lanese, C.A. Mezzina, and F. Tiezzi. Causal-consistent rollback in a tuple-based language. J. Log. Algebr. Meth. Program. Volume 88, pages 99–120. 2017.

[32] Xiaohu Ge, Jing Yang, Hamid Gharavi, and Yang Sun. Energy Efficiency Challenges of 5g Small Cell Networks. *arXiv:1702.03503 [cs]*, February 2017. arXiv: 1702.03503.

[33] J. Hoydis, S. ten Brink, and M. Debbah. Massive MIMO in the UL/DL of Cellular Networks: How Many Antennas Do We Need? *IEEE Journal on Selected Areas in Communications*, 31(2):160–171, February 2013.

[34] A. O. Holder and C. D. Carothers. Analysis of time warp on a 32,768 processor IBM Blue Gene/L supercomputer. In Proceedings of the European Modeling and Simulation Symposium (EMSS), 2008.

[35] H. Hamann, T. Schmickl, and K. Crailsheim. Thermodynamics of emergence: Langton's ant meets Boltzmann. In *2011 IEEE Symposium on Artificial Life (ALIFE)*, pages 62–69, April 2011.

[36] J. Conrod. Tutorial: Reverse debugging with GDB 7, 2009 `http://jayconrod.com/posts/28/tutorial-reverse-debugging-with-gdb-7`

[37] S. T. King, G. W. Dunlap, and P. M. Chen. Debugging operating systems with time-traveling virtual machines. In USENIX Annual Technical Conference, General Track, pages 1-15, 2005.

[38] Kollmitzer, Christian, et al. "Applied quantum cryptography". Vol. 797. Springer, 2010.

[39] S. Kuhn, and I. Ulidowski. Towards modelling of local reversibility. In Proceedings of Reversible Computation 2015, pages 279–284. LNCS 9138. Springer, 2015.

[40] S. Kuhn, and I. Ulidowski. A Calculus for Local Reversibility. In Proceedings of Reversible Computation 2016, pages 20–35. LNCS 9720. Springer, 2016.

[41] S. Kuhn, and I. Ulidowski. Local reversibility in a Calculus of Covalent Bonding. *Science of Computer Programming*, 151:18–47, January 2018.

[42] Jia Lee, Susumu Adachi, Yun-Ni Xia, and Qing-Sheng Zhu. Emergence of universal global behavior from reversible local transitions in asynchronous systems. *Information Sciences*, 282:38–56, October 2014.

[43] J. M. LaPre, E. J. Gonsiorowski, and C. D. Carothers. Lorain: A step closer to the pdes 'holy grail'. In Proceedings of the 2Nd ACM SIGSIM/PADS Conference on Principles of Advanced Discrete Simulation, SIGSIM-PADS '14, pages 3–14, New York, NY, USA, 2014. ACM.

[44] K. Lee, S. Lee, and H. Yoo. Low-power network-on-chip for high-performance soc design. *IEEE Trans. VLSI Syst.*, 14(2):148–160, 2006.

[45] I. Lanese, C.A. Mezzina, A. Schmitt, and J-B. Stefani. Controlling reversibility in Higher-Order Pi. In Proceedings of CONCUR 2011, pages 297-311, volume 6901 of Lecture Notes

[46] I. Lanese, C. A. Mezzina, F. Tiezzi: Causal-Consistent Reversibility. Bulletin of the EATCS 114 (2014)

[47] Shan Lu, Soyeon Park, Eunsoo Seo, Yuanyuan Zhou: Learning from mistakes: a comprehensive study on real world concurrency bug characteristics. ASPLOS 2008: 329-339

[48] J.S. Laursen, U.P. Schultz, and L.-P. Ellekilde, "Modelling Reversible Execution of Robotic Assembly". Accepted for publication in Robotica.

[49] J.S. Laursen, U.P. Schultz, and L.-P. Ellekilde, "Automatic error recovery in robot assembly operations using reverse execution," in International Conference on Intelligent Robots and Systems (IROS 2015), IEEE/RSJ, 2015.

[50] A. Mammela and A. Anttonen. Why Will Computing Power Need Particular Attention in Future Wireless Devices? *IEEE Circuits and Systems Magazine*, 17(1):12–26, 2017.

[51] Henrik Mhe and Andreas Angerer and Alwin Hoffmann and Wolfgang Reif. On reverse-engineering the KUKA Robot Language, 1st International Workshop on Domain-Specific Languages and models for ROBotic systems. 2010.

[52] T. L. Marzetta. Noncooperative Cellular Wireless with Unlimited Numbers of Base Station Antennas. *IEEE Transactions on Wireless Communications*, 9(11):3590–3600, November 2010.

[53] Mehic, Miralem, et al. "Calculation of the key length for quantum key distribution." Elektronika ir Elektrotechnika 21.6 (2015): 81-85.

[54] Mehic, Miralem, et al. "Calculation of key reduction for B92 QKD protocol." Quantum Information and Computation XIII. Vol. 9500. International Society for Optics and Photonics, 2015.

[55] Mehic, Miralem, et al. "Analysis of the public channel of quantum key distribution link." IEEE Journal of Quantum Electronics 53.5 (2017): 1-8.

[56] Mehic, Miralem, et al. "Implementation of quantum key distribution network simulation module in the network simulator NS-3." Quantum Information Processing 16.10 (2017): 253.

[57] Irene Macaluso, Carlo Galiotto, Nicola Marchetti, and Linda Doyle. A complex systems science perspective on wireless networks. *Journal of Systems Science and Complexity*, 29(4):1034–1056, August 2016.

[58] Mink, Alan, et al. "QKD on a board limited by detector rates in a free-space environment." Proc. 7th Int. Conf. Quantum, Nano Micro Technol.(ICQNM). 2013.

[59] Kenichi Morita. Reversible computing and cellular automata—A survey. *Theoretical Computer Science*, 395(1):101–131, April 2008.

[60] Nguyen, Kim-Chi, et al. "Side-information coding with turbo codes and its application to quantum key distribution." arXiv preprint cs/0406001 (2004).

[61] Naoki Nishida, Adrián Palacios, Germán Vidal. A Reversible Semantics for Erlang. Proc. of the 26th International Symposium on Logic-Based Program Synthesis and Transformation, LOPSTR 2016. LNCS 10184, Springer. To appear.

[62] Rumyana Neykova and Nobuko Yoshida. Let It Recover: Multiparty Protocol-Induced Recovery. 26th International Conference on Compiler Construction (CC), pages 98–108. ACM. 2017.

[63] A. G. Ortiz, L. S. Indrusiak, T. Murgan, and M. Glesner. Low-power coding for networks-on-chip with virtual channels. *J. Low Power Electronics*, 5(1):77–84, 2009.

[64] P. R. Panda and N. D. Dutt. Reducing address bus transitions for low power memory mapping. In *European Design and Test Conference, ED&TC*, pages 63–71, 1996.

[65] Perumalla, K.S.: Introduction to Reversible Computing, Chapman and Hall/CRC, 325 Pages, ISBN 9781439873403, 2013.

[66] K. S. Perumalla and A. J. Park. Reverse computation for rollback-based fault tolerance in large parallel systems. Cluster Computing, 17(2):303–313, June 2014.

[67] I.C.C. Phillips and I. Ulidowski. Reversing algebraic process calculi. Journal of Logic and Algebraic Programming, 73:7096, 2007.

[68] Gh. Paun. Computing with Membranes, Journal of Computer and System Sciences, 61(1): 108-143, 2000.

[69] A. Pellegrini, R. Vitali, and F. Quaglia. Autonomic state management for optimistic simulation platforms. IEEE Transactions on Parallel and Distributed Systems, 26(6):1560–1569, June 2015.

[70] Renner, Renato. "Security of quantum key distribution." International Journal of Quantum Information 6.01 (2008): 1-127.

[71] S. Ramprasad, N. R. Shanbhag, and I. N. Hajj. A coding framework for low-power address and data busses. *IEEE Trans. VLSI Syst.*, 7(2):212–221, 1999.

[72] Schultz, Ulrik, Mirko Bordignon, and Kasper Stoy. "Robust and reversible execution of self-reconfiguration sequences." Robotica 29.01 (2011): 35-57.

[73] Sugimoto, Tomohiro, et al. "A study on secret key reconciliation protocol." IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences 83.10 (2000): 1987-1991.

[74] E. Santini, M. Ianni, A. Pellegrini, and F. Quaglia. Hardware-transactional-memory based speculative parallel discrete event simulation of very ne grain models. In 2015 IEEE 22nd International Conference on High Performance Computing (HiPC), pages 145-154, Dec 2v015.

[75] Reverse Code Generation for Parallel Discrete Event Simulation. Markus Schordan, David Jefferson, Peter Barnes, Tomas Oppelstrup, Daniel Quinlan. In Proceedings of the 7th Conference on Reversible Computation. Jean Krivine and Jean-Bernard Stefani (Eds.): Reversible Computation, LNCS 9138, pp. 95-110, ISBN 978-3-319-20859-6, Springer, 2015.

[76] U.P. Schultz, J.S. Laursen, L.-P. Ellekilde, and H.B. Axelsen, "Towards a domain-specific language for reversible assembly sequence," in Reversible Computation, 2015.

[77] Schultz, Ulrik. "Towards a general-purpose, reversible language for controlling self-reconfigurable robots." Reversible Computation. Springer Berlin Heidelberg, 2012. 97-111.

[78] Scarani, Valerio, et al. "The security of practical quantum key distribution." Reviews of modern physics 81.3 (2009): 1301.

[79] Markus Schordan, Tomas Oppelstrup, David Jefferson, Peter D. Barnes Jr., Daniel J Quinlan. Automatic Generation of Reversible C++ Code and Its Performance in a Scalable Kinetic Monte-Carlo Application. SIGSIM-PADS 2016: 111-122, 2016.

[80] Overview of the TPC-C Benchmark. `http://www.tpc.org/tpcc/detail.asp`

[81] Mobile Robot API for Janus. Master thesis defence by Martin Nyborg Thomsen. University of Copenhagen, Denmark, 2014.

[82] UndoDB. Commercial debugger `http://undo-software.com/products/undodb/`

[83] Undo Software. Increasing software development productivity with reversible debugging, 2014. in Computer Science, pages 297311. Springer, 2011.

[84] P. Viswanath and D. N. C. Tse. Sum capacity of the vector Gaussian broadcast channel and uplink-downlink duality. *IEEE Transactions on Information Theory*, 49(8):1912–1921, August 2003.

[85] R. Wille, R. Drechsler, C. Osewold, and A. G. Ortiz. Automatic design of low-power encoders using reversible circuit synthesis. In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1036–1041, 2012.

[86] R. Wille, O. Keszocze, S. Hillmich, M. Walter, and A. G. Ortiz. Synthesis of approximate coders for on-chip interconnects using reversible logic. In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1140–1143, 2016.

[87] A. Zulehner and R. Wille. Taking one-to-one mappings for granted: Advanced logic design of encoder circuits. In *Design, Automation and Test in Europe*, 2017.

[88] Tetsuo Yokoyama, Holger Bock Axelsen, and Robert Glck. 2008. Principles of a reversible programming language. In Proceedings of the 5th conference on Computing frontiers (CF '08). ACM, New York, NY, USA, 43-54.

[89] QISKIT Python SDK. `https://github.com/QISKit/qiskit-sdk-py`. Accessed: 2017-09-15.

[90] M. Amy, D. Maslov, M. Mosca, and M. Roetteler. A meet-in-the-middle algorithm for fast synthesis of depth-optimal quantum circuits. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 32(6):818–830, 2013.

[91] I. Cassar, A. Francalanza, C. A. Mezzina, and E. Tuosto. Reliability and fault-tolerance by choreographic design. In A. Francalanza and G. J. Pace, editors, *Proceedings Second International Workshop on Pre- and Post-Deployment Verification Techniques, PrePost@iFM 2017, Torino, Italy, 19 September 2017.*, volume 254 of *EPTCS*, pages 69–80, 2017.

[92] A. Francalanza, C. Antares, and E. Tuosto. Reversible choreographies via monitoring in erlang. In S. Bonomi and E. Rivire, editors, *Distributed Applications and Interoperable Systems - 18th IFIP WG 6.1 International Conference, DAIS 2018*, Lecture Notes in Computer Science. Springer, 2018. to appear.

[93] A. S. Green, P. L. Lumsdaine, N. J. Ross, P. Selinger, and B. Valiron. Quipper: a scalable quantum programming language. In *Conference on Programming Language Design and Implementation*, pages 333–342, 2013.

[94] T. Häner, D. S. Steiger, M. Smelyanskiy, and M. Troyer. High performance emulation of quantum circuits. In *International Conference for High Performance Computing, Networking, Storage and Analysis*, page 74, 2016.

[95] H. Hiraishi and H. Imai. BDD operations for quantum graph states. In *Reversible Computation - 6th International Conference, RC 2014, Kyoto, Japan, July 10-11, 2014. Proceedings*, pages 216–229, 2014.

[96] N. Khammassi, I. Ashraf, X. Fu, C. G. Almudéver, and K. Bertels. QX: A high-performance quantum computer simulation platform. In *Design, Automation & Test in Europe Conference & Exhibition, DATE 2017, Lausanne, Switzerland, March 27-31, 2017*, pages 464–469, 2017.

[97] I. Lanese, N. Nishida, A. Palacios, and G. Vidal. CauDEr website. URL: `https://github.com/mistupv/cauder`.

[98] I. Lanese, N. Nishida, A. Palacios, and G. Vidal. CauDEr: A Causal-Consistent Reversible Debugger for Erlang. In *14th International Symposium on Functional and Logic Programming (FLOPS'18)*, volume 10818 of *LNCS*. Springer, 2018. To appear.

[99] K. Matsumoto and K. Amano. Representation of quantum circuits with clifford and $\pi/8$ gates. *arXiv preprint arXiv:0806.3834*, 2008.

[100] D. M. Miller, R. Wille, and Z. Sasanian. Elementary quantum gate realizations for multiple-control toffoli gates. In *41st IEEE International Symposium on Multiple-Valued Logic, ISMVL 2011, Tuusula, Finland, May 23-25, 2011*, pages 288–293, 2011.

[101] P. Niemann, R. Wille, D. M. Miller, M. A. Thornton, and R. Drechsler. Qmdds: Efficient quantum function representation and manipulation. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 35(1):86–99, 2016.

[102] E. Pednault, J. A. Gunnels, G. Nannicini, L. Horesh, T. Magerlein, E. Solomonik, and R. Wisnieff. Breaking the 49-qubit barrier in the simulation of quantum circuits. *arXiv preprint arXiv:1710.05867*, 2017.

[103] M. Saeedi, R. Wille, and R. Drechsler. Synthesis of quantum circuits for linear nearest neighbor architectures. *Quantum Information Processing*, 10(3):355–377, 2011.

[104] V. Samoladas. Improved BDD algorithms for the simulation of quantum circuits. In *European Symposium on Algorithms*, pages 720–731, 2008.

[105] A. Shafaei, M. Saeedi, and M. Pedram. Optimization of quantum circuits for interaction distance in linear nearest neighbor architectures. In *The 50th Annual Design Automation Conference 2013, DAC '13, Austin, TX, USA, May 29 - June 07, 2013*, pages 41:1–41:6, 2013.

[106] A. Shafaei, M. Saeedi, and M. Pedram. Qubit placement to minimize communication overhead in 2d quantum architectures. In *19th Asia and South Pacific Design Automation Conference, ASP-DAC 2014, Singapore, January 20-23, 2014*, pages 495–500, 2014.

[107] M. Smelyanskiy, N. P. D. Sawaya, and A. Aspuru-Guzik. qHiPSTER: The quantum high performance software testing environment. *CoRR*, abs/1601.07195, 2016.

[108] D. S. Steiger, T. Häner, and M. Troyer. ProjectQ: an open source software framework for quantum computing. *arXiv preprint arXiv:1612.08091*, 2018.

[109] G. F. Viamontes, I. L. Markov, and J. P. Hayes. *Quantum Circuit Simulation*. Springer, 2009.

[110] D. Wecker and K. M. Svore. LIQUi|>: A software design architecture and domain-specific language for quantum computing. *CoRR*, abs/1402.4467, 2014.

[111] R. Wille, O. Keszöcze, M. Walter, P. Rohrs, A. Chattopadhyay, and R. Drechsler. Look-ahead schemes for nearest neighbor optimization of 1d and 2d quantum circuits. In *21st Asia and South Pacific Design Automation Conference, ASP-DAC 2016, Macao, Macao, January 25-28, 2016*, pages 292–297, 2016.

[112] R. Wille, A. Lye, and R. Drechsler. Exact reordering of circuit lines for nearest neighbor quantum architectures. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 33(12):1818–1831, 2014.

[113] A. Zulehner, S. Gasser, and R. Wille. Exact global reordering for nearest neighbor quantum circuits using A^*. In *International Conference on Reversible Computation*, pages 185–201. Springer, 2017.

[114] A. Zulehner, A. Paler, and R. Wille. Efficient mapping of quantum circuits to the IBM QX architectures. In *2018 Design, Automation & Test in Europe Conference & Exhibition, DATE 2018, Dresden, Germany, March 19-23, 2018*, pages 1135–1138, 2018.

[115] A. Zulehner and R. Wille. Advanced simulation of quantum computations. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2018.