FPGA Accelerated Analysis of Boolean Gene Regulatory Networks

Matteo Manica, Raphael Polig, Mitra Purandare, Roland Mathis, Christoph Hagleitner, and María Rodríguez Martínez

Abstract—Boolean models are a powerful abstraction for qualitative modeling of gene regulatory networks. With the recent availability of advanced high-throughput technologies, Boolean models have increasingly grown in size and complexity, posing a challenge for existing software simulation tools that have not scaled at the same speed. Field Programmable Gate Arrays (FPGAs) are powerful reconfigurable integrated circuits that can offer massive performance improvements. Due to their highly parallel nature, FPGAs are well suited to simulate complex molecular networks. We present here a new simulation framework for Boolean models, which first converts the model to Verilog, a standardized hardware description language, and then connects it to an execution core that runs on an FPGA coherently attached to a POWER8 processor. We report an order of magnitude speedup over a multi-threaded software simulation tool running on the same processor on a selection of Boolean models. Analysis on a T-cell large granular lymphocyte leukemia (T-LGL) demonstrates that our framework achieves consistent performance improvements resulting in new biological insights. In addition, we show that our solution allows to perform attractor detection at an unprecedented speed, exhibiting a speedup ranging from one to three orders of magnitude compared to alternative software solutions.

Index Terms—Field programmable gate arrays, accelerator architectures, mathematical model, biological system modeling, biological systems, biological control systems, biological processes, computer simulation, systems simulation, computational biology, computational systems biology, bioinformatics, systems biology

1 Introduction

Genes do not work in isolation, but exert their function in complex and tightly connected gene regulatory networks (GRNs) [1]. At the very basis, understanding complex diseases amounts to unravelling normal and dysregulated behavior of GRNs. However, due to their complexity and the lack of quantitative knowledge about most kinetic parameters governing molecular interactions, an exact analysis of GRNs, usually based on ordinary differential equations, is in most cases not possible.

Boolean models [2] are an attractive alternative approach for the study of GRNs that are consistently used in the systems biology community [3], [4], [5], [6], [7], [8], [9], [10], [11]. Boolean models provide a qualitative description of a GRN, where chemical species concentrations or activities are represented using a finite set of discrete values. In a Boolean model, a node corresponds to a species, e.g., a gene, and an edge represents an interaction between species. In its simplest form, a gene can be ON (1) or OFF (0), and its interactions with other genes are defined by means of a Boolean function of its immediate parent nodes in the GRN. Time is represented by discrete steps, after which the Boolean functions are evaluated following an update scheme and the new values are assigned to their corresponding genes [12]. Various update schemes can be adopted. In the *synchronous* scheme [13], all genes are simultaneously evaluated and updated, resulting in a fully deterministic and computationally tractable system, although often biologically unrealistic. An asynchronous scheme [14] takes in account time diversity associated with the different reaction rates of biological systems by updating variables in a non-synchronous order. Multiple asynchronous update schemes are possible, e.g., deterministic asynchronous, stochastic asynchronous, random asynchronous, etc. [15]. We focus on a *random asynchronous* update scheme in which a gene is chosen randomly and updated to its next value. The *asynchronous* scheme provides a stochastic, and hence more realistic, description of a GRN, although at the price of greatly increasing the computational complexity and execution time of the model. In addition, as the model is stochastic, it has to be run multiple times in order to resolve the mean dynamical behavior, resulting often in prohibitively long simulation times.

Although a Boolean model cannot provide the level of detailed information that an experimentally well-characterized ordinary differential equation system can achieve, it can produce a qualitative description of the most salient features of a dynamical system. For instance, Boolean models can be used to identify steady states, cyclic states or attractors – cycles of states A such that no trajectory entering A can leave A, see for instance [16], [17], [18]. Attractors, in particular, can provide valuable information about the observed phenotypes and underlying mechanisms associated with complex diseases, such as cancer [19]. However, the problem of finding attractors is characterized by a high computational complexity, which steeply increases with the number of network nodes. Furthermore, the number and the size of the attractors of a system changes dramatically with the update scheme [20]. Some types of attractors, such as self-loops and simple loops, are common to both update schemes and hence can be computed using the less expensive synchronous update scheme. However, in the most general case, the characterization of the attractors landscape of a model requires asynchronous updates, resulting in high complexity in the number of states forming the attractor, as well as lengthy transitory states leading to an attractor [12].

The computational problem of finding all the attractors in a Boolean model is extremely hard. Even the simpler problem of finding the steady states in a Boolean model is NP-hard [21], [22], indicating that it is not possible to efficiently, i.e., in polynomial time, find all attractors in the analysed system. Some of our coauthors [15] have proposed a fast and scalable solution using FPGAs to simulate Boolean models. The framework supports synchronous and asynchronous updates. The approach scales efficiently, showing a significant speedup compared with BoolNet [23], a popular R package for the construction and analysis of Boolean networks. In this paper, we extend our FPGA simulator to detection of attractors leveraging the highly parallel nature and ever incresing capacity of FPGAs for attractor detection.

Our accelerator is seamlessly integrated with a POWER8 processor, greatly increasing the usability of the proposed framework. We demonstrate the performance of our accelerator using six state-of-the-art Boolean models from the literature, including models for T-cell large granular lymphocyte leukemia [24], castration resistant prostate cancer [6], signaling pathways involved in cancer [7], colon cancer [4], Fanconi anemia and breast cancer [25], and the MAPK pathway [8]. First, using the 3 largest models, we compare the runtime performance of our framework with a multi-threaded implementations of two commonly used software tools, namely BoolNet [23] and BooleanNet [26], both running on a POWER8 processor, and with an existing accelerator proposed by Miskov-Zivanov et al. [27]. Our solution demonstrates an order of magnitude speedup over BoolNet, which already runs significantly faster than BooleanNet, and exhibits better performance than the Miskov-

- *R. Polig, M. Purandare, R. Mathis, C. Hagleitner, and M.R. Martínez are with IBM Research Zürich, Rüschlikon 8803, Switzerland. E-mail: {pol, mpu, lth, hle, mrm}@zurich.ibm.com.*
- *M. Manica with IBM Research Zürich, Rüschlikon 8803, Switzerland, and also with ETH Zürich, Zürich 8092, Switzerland. E-mail: tte@zurich.ibm.com.*

Manuscript received 21 Nov. 2018; revised 29 July 2019; accepted 6 Aug. 2019. Date of publication 3 Sept. 2019; date of current version 8 Dec. 2020.
(Corresponding author: María Rodríguez Martínez.)
Digital Object Identifier no. 10.1109/TCBB.2019.2936836

Zivanov accelerator. Second, we include an analysis of the dynamic behavior of some key signaling pathways in the large granular lymphocyte leukemia (T-LGL) model [24]. Lastly, we apply our framework to attractor analysis in all the six models considered. Our accelerator reaches a speedup of one to three orders of magnitude over BoolNet and demonstrates consistent advantages when compared to symbolic approaches for attractor detection [28], [29].

2 Related Work

Simulators like BooleanNet [26] and BoolNet [23] analyze Boolean GRNs. However, simulations on conventional computers, especially, using asynchronous updates, usually result in prohibitively long execution times due to the intrinsic disparity between the sequential steps executed by a microprocessor program and the highly parallel nature of biological systems [30]. Stochastic simulators have also been proposed. For instance, MaBoSS [31] simulates individual time trajectories using a Monte-Carlo kinetic algorithm (or Gillespie algorithm) and provides a generalization of the asynchronous Boolean dynamical rules. Similarly to conventional simulators, stochastic simulators suffer from long execution times.

Most common methods to compute attractors start with randomly selected initial states and perform exhaustive searches of the state space of a network. However, the time complexity of these methods grows exponentially with the number of nodes in the network, and hence, techniques to alleviate the complexity of the state space are needed. For instance, the entire network state space can be appropriately broken down into selected subspaces that can be exhaustively searched [32]. However, this approach is not scalable and it is currently limited to networks of up to 150 nodes. Network reduction techniques that conserve the fixed points and complex attractors of asynchronous Boolean models have also been developed [33]. In a different approach, a systematic removal of state transitions renders the state transition graph acyclic and transforms all attractors into fixed points that can be enumerated with little effort [20]. A mathematical model of a pruned portion of the state space, followed by a randomized traversal method to extract the steady states in the remaining state space, has also been proposed to increase speed and scalability [34]. Finally, variants of the Gillespie algorithm have also been used to compute probability estimates of attractor reachability in asynchronous dynamics [35].

When approximate solutions are not desirable, symbolic approaches can be efficient as they do not perform explicit traversal of the state space. Reduced ordered binary decision diagrams (ROBDDs) use directed acyclic graphs to represent large Boolean functions in a space-efficient manner, and are computationally suitable for complex Boolean operations, e.g., logical AND, OR, etc, and set operations, e.g., union, intersection, etc. Some of the tools that use ROBDDs are *geneFAtt* [36] and *boolSim*/*genYsis* [28]. A decomposition method based on strongly connected components is proposed in [37]. However, binary decision diagrams (BDDs) have generally unpredictable memory requirements. Satisfiability solvers, usually more scalable than BDDs, are also popular in attractor computation [38], [39]. But with increasing number of genes and length of Boolean rule unwinding, these approaches become inefficient. Analysis of Networks through TEmporal-LOgic sPEcifications (Antelope) uses model checkers, a collection of techniques for automatically verifying properties of discrete systems, and for analyzing and constructing Boolean GRNs [40]. Unlike simulators, model checkers can prove properties of a set of infinitely many paths. In addition, they can handle new, unforeseen properties by simply adding temporal-logic formulas, while simulators require the incorporation of such properties in their program code. Despite these properties, one common disadvantage of symbolic approaches with respect to explicit approaches is that attractors are available only at the very end of the computation, which can take a prohibitively long time and possibly, large memory.

While explicit approaches are not scalable, they present results as and when available. A practical solution is to accelerate them using highly parallel Field programmable Gate Arrays (FPGAs). A handful of hardware accelerated biological network simulators have been proposed in the past. A mix of digital and large-signal analog computation has been proposed for the simulation of gene regulatory networks [41]. It is reported to simulate networks of up to 20 nodes. FPGA-accelerated attractor computation of scale-free gene regulatory networks is proposed in [42], [43], [44]. Some [30], [45] implement variants of Gillespie's stochastic simulation algorithm on FPGAs. They have demonstrated the suitability of FPGA technology for the simulation of variants of the Gillespie algorithm, achieving a performance 20 times faster than a competing general purpose CPU. An FPGA-based accelerator framework for Boolean models has been demonstrated by Miskov-Zivanov et al. [27], [46]. While this framework supports asynchronous simulation, it does not perform attractor analysis. Also, the framework is not fully integrated with the host system, limiting its accessibility by the user software. For instance, buttons are used to manually start and stop the simulation on the FPGA, and the state of the network is displayed using 7-segment LED displays. This prohibits any further analysis of computed results. More recently, da Silva et al. [47] presented an integrated acceleration framework for synchronous GRNs using a tightly coupled architecture on an Intel Xeon processor and an Intel Stratix V FPGA. It provides up to two orders of magnitude speedup over a parallel OpenMP implementation.

Input GRN model format: The tools for simulation and analysis of Boolean GRNs unfortunately do not agree on a common input format. This has been a main hindrance to comparing our work to others. The SAT-based tool from Dubrova et al. [38] accepts the models in Berkeley Logic Interchange Format (BLIF), which is specific to the field of electronic design automation. Boolnet and Booleannet do not agree on the input model format as well. There are ongoing efforts towards standardization using a common model representation format such as SBML-qual [48].

Our work presents an FPGA-accelerated framework for the simulation and analysis of Boolean networks that can also identify synchronous attractors. Our proposed approach is seamlessly integrated with a POWER8 processor, which greatly increases its usability and integration capabilities with other software tools.

3 Methods

This section describes our accelerator framework detailing its architecture and system integration.

Host Processor and FPGA Integration. The host is an IBM POWER8-based server system with the ability to coherently connect an FPGA via the coherent accelerator processor interface (CAPI). This enables the FPGA to act as a part of a software process and access virtual memory locations just like a regular processor core. Also, it allows the FPGA to access the system's main memory that has been allocated by the software process owning the accelerator. The proposed solution allows a seamless integration of the FPGA and the host processor. Fig. 1 provides an overview of the overall system architecture.

Input Arguments. The end-user is required to provide six arguments: i) a Boolean model definition; ii) an update order, i.e., synchronous or asynchronous update; iii) a number of simulation repetitions if an asynchronous update scheme is selected; iv) the list of start states to analyze; v) the number of time steps to simulate, and vi) a flag indicating whether to perform attractor analysis.

Detailed Hardware Acceleration Process. Once the arguments are received, our framework performs the following steps. First, the host converts the Boolean model into a hardware description language (HDL) model. Verilog is the chosen HDL. Second, the host creates a bit stream and configures the FPGA card for the computation. Afterwards, the simulation parameters are transferred from

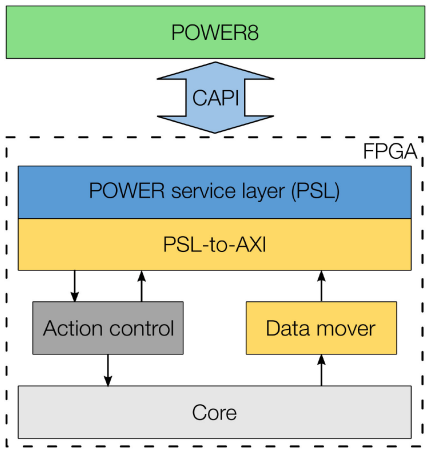

Fig. 1. *System architecture overview.* Overall system architecture with the FPGA top-level. Communication between the FPGA card and the POWER8 processor is performed through CAPI.

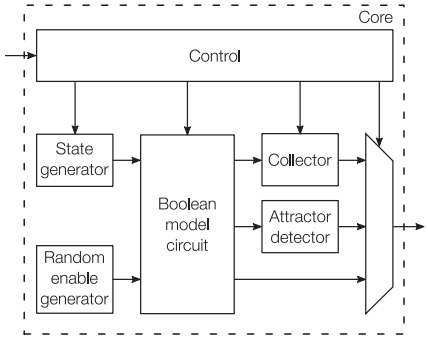

Fig. 2. *Execution core scheme.* The top-level modules are used in the execution core to implement: synchronous and asynchronous simulations as well as attractor detection.

the host to the FPGA. The simulation is then started on the FPGA using the chosen update strategy and the given start states. Optionally, if enabled, the FPGA checks for attractors. As soon as the results are processed the FPGA reports them to the host. Once the results are in the host, they can be either displayed via a graphical user interface or written on disk for further analysis.

Execution Core. As depicted in Fig. 1, we have put two types of modules on the FPGA, namely, the core module and the communicating module (to-and-from host). The core module contains the Boolean model and is responsible for simulation and analysis. The remaining modules for POWER service layer, PSL-to-AXI etc. form the communicating module. Fig. 2 illustrates the top-level of the core with its main components.

The Boolean model is embedded in the Boolean Model Circuit (BMC). In addition, the execution core contains all the necessary components to perform simulations of the Boolean model and further analyze the results. The core receives a *start* signal together with a set of arguments listed before. The implemented computational core is capable of performing synchronous or asynchronous simulations of the Boolean model and can detect simple attractors for synchronous updates. The random enable generator block (bottom left part of Fig. 2) takes care of selecting the node update order accordingly.

In the asynchronous mode, a simulation is run for a given number of time steps for all the start states provided by the user, i.e., for all the start states, the simulations are repeated for a user-specified number of times. The core captures the states reached in the multiple simulation iterations.

Synchronous Attractor Computation. In the synchronous mode, the execution core can also perform an exhaustive search for attractors. The attractor detection module stores all visited states during a simulation in a local time series memory block. Before the current state is added to the list, the core checks if the state is present in the time series list. If the state is not present, it is added to the list. Otherwise, the simulation stops and the attractor states are copied to a local attractor list memory block. To identify the attractors, pointers to the start state of an attractor are stored in a third memory block. The core then moves on to the next initial state supplied to it.

In the current implementation, the attractor module is disabled for asynchronous updates. If the attractor module is activated for asynchronous updates, simulations must continue even if a state is visited more than once. In this case, the attractor module will detect all the cycles. Additional checks/computations are needed to find attractors arising from the detected cycles.

Reporting Asynchronous Simulation Results: Due to the deterministic nature of synchronous updates, a state of a Boolean model has only one successor state. For a given input, the value of a particular output at a particular time step remains the same across all simulation repetitions. Hence, it is feasible to report all the states reached during simulation to the host/software.

This is not the case for asynchronous updates. For a fixed input and two simulation repetitions, the generated sequences of random permutations/updates is potentially different. Different sequences of update orders most likely result in different outputs. Hence, outputs at the same time step can be different for different simulation iterations. We present the results of such simulations in a meaningful manner. The collector module records how often a particular node is active at a given time step. For all the nodes and all the time steps, the fraction of simulations in which the gene node is activated (node set to ON) at that time step is computed. The fraction of simulations is then divided by the total number of simulation repetitions. This gives us the *activation frequency* of each node at each time step.

4 Results

The performance of our accelerator framework is evaluated on six published models with varied number of nodes and complexity: i) T-LGL, a Boolean model proposed in [24] for T-cell large granular lymphocyte leukemia (model version from the set of examples that are provided with BooleanNet); ii) CRPC, a model by Hu et. al [6] that includes relevant pathways for castration resistant prostate cancer; iii) Fumia, a model that incorporates the main signaling pathways in cancer [7]; iv) CAC, a model for the development of colitis-associated colon cancer that integrates the extracellular environment and intracellular signalling pathways [4]; v) FA-BRCA, a Boolean model of Fanconi Anemia/Breast Cancer (FA/BRCA) pathway [25]; and vi) MAPK, a comprehensive model of MAPK pathway [8].

4.1 Asynchronous Simulation

Runtime Analysis. Software simulations of BoolNet and BooleanNet constitute the baseline and are performed on the same POWER8-based server node that hosts the FPGA accelerator. The server has 20 physical cores running at 2.29 GHz and a total of 512 GB DDR3 RAM. The simulations are run using the BooleanNet Python package and the BoolNet R package. The benchmarks processed all simulation jobs with 20 worker threads simultaneously to fully utilize the server node.

Our framework uses the Xilinx Kintex UltraScale KU060 FPGA and the target frequency is 250 MHz. The measurements include the time for transferring the parameters to the FPGA and transferring the results from the FPGA to the main memory. Only one software thread has been used to perform the memory management and control for the FPGA.

We ran our accelerator for asynchronous simulations on the following models: T-LGL, CRPC and Fumia. Table 1 summarizes the

TABLE 1
Asynchronous Simulation Benchmark

Model	# input	# output	# sim.	Time B1	Time B2	Time FPGA	Time B1	Time B2	Time FPGA
T-LGL	4	47	16	90.1s	0.88s	0.12s	5.6s	0.043s	0.007s
CRPC	22	69	64	580.5s	3.51s	0.23s	9.1s	0.043s	0.003s
Fumia	6	92	64	7895.7s	3.52s	0.30s	123.4s	0.044s	0.004s

Summary of the total execution times for evaluated models: T-LGL, CRPC and Fumia. The total running times of 100 repetitions with 100 time steps each in the asynchronous mode are reported.

results for each model. The first two columns list the number of inputs and outputs. All possible input combinations are generated as individual simulation jobs. These are listed in the third column labelled #sim. in Table 1). Each job is then simulated by BooleanNet, BoolNet, and the FPGA. The next thre columns report the runtimes of each of these tools. Only for the CRPC model the number of simulations has been limited due to the long runtime. Each simulation job has been simulated in asynchronous mode on the models for 100 time steps and repeated 100 times. The last three columns report time taken by BooleanNet, BoolNet, and the FPGA for a single simulation, i.e., runtime divided by the number of simulations.

Compared to BooleanNet and BoolNet, the FPGA accelerator exhibits a speedup of 750.8x and 7.3x respectively for the T-LGL model. For the CRPC model, it takes a prohibitively long time to generate all inputs in case of software simulations and hence, the number of simulations is limited to 64. While BooleanNet apparently struggles to simulate the CRPC model, the runtime of BoolNet is dominated by the number of simulations. In this case the speedup obtained is 2523.9x compared to BooleanNet and 15.2x compared to BoolNet. For the Fumia model, the FPGA accelerator demonstrates a speedup of 26,319x and 11.7x over BooleanNet and BoolNet, respectively.

Comparison with Miskov-Zivanov et al. [27]. The FPGA framework presented in Miskov-Zivanov et al. [27] reports an asynchronous simulation time of 0.019s on the T-LGL model for 200 repeats and 15 time steps. These experiments have been conducted on a standalone FPGA board at 50 MHz. Adjusting this number for 100 time steps, 100 iterations, and a frequency of 250 MHz, such a simulation would take 0.012s. This is 68 percent slower than the asynchronous simulations performed by the presented architecture, where the deteriorated performance is mainly due to the generation of the random update order. Specifically, the slow speed arises from its reliance on the random order generated by the linear-feedback shift register (LFSR). Note that the LFSRs also lead to a non-deterministic runtime of the architecture.

Dynamic Behavior Analysis of T-LGL. Analysis of the dynamic behavior of the T-LGL leukemia network using asynchronous simulations identified a diverging dynamics associated with the apoptosis, i.e., programmed cell death, output node [49]. Namely, when the node is ON, a single steady state associated with apoptosis is found. Conversely, when the node is stabilized at OFF, two additional fixed points for which the cells escape apoptosis are found. This criterion can be used to group steady state behavior into the T-LGL leukemia class (diseased state) and into the apoptosis attractor class (normal state), showing the importance of describing accurately network dynamics.

We compute the activation frequencies of all the nodes in the T-LGL model and study model dynamics and the reached steady states. Activation frequencies for representative nodes, e.g., apoptosis node and BID (BH3 Interacting Domain Death Agonist) node, whose over-expression was predicted to lead to apoptosis in T-LGL cells [49], are shown in Fig. 3. It is evident from the figure that low number of repetitions (<1000) results in curves that are

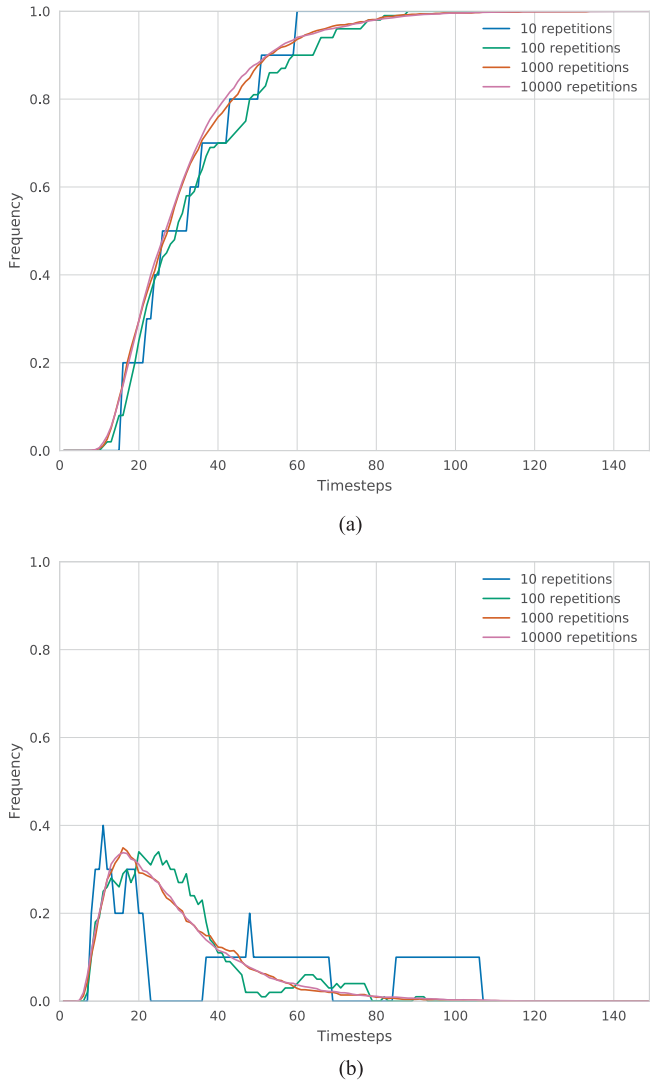

Fig. 3. *Frequency at different number of repetitions.* Average activation frequencies over time for the apoptosis (a) and BID (b) nodes. The increased number of repetitions results in smoother curves that capture better the dynamics of the system and are more consisting with the biological expected behavior.

not smooth, rendering difficulties in the accurate prediction of the biological properties of the system. As the number of repetitions is increased, the curves become smoother and more consistent with the expected behavior. The order of magnitude speedup achieved by our hardware accelerator framework enables larger number of simulation repetitions for the same initial state, resulting in smoother curves that better capture the dynamical evolution of the network with time. Accurate dynamic estimation of the current state of a system also plays a critical role in attractor analysis, as convergence of node activation frequencies is an indicator of the presence of an attractor. Fig. 3 shows how increasing the number of repetitions for a given initial state changes the activation frequency estimates in the T-LGL model. The average curves for the different repetitions numbers converge to smooth curves around 1000 repetitions. The curves for the apoptosis node illustrate how the estimates change their evolution over the time steps: fewer repetitions underestimate the steepness, while higher numbers of repetitions capture the dynamics of the system in a more consistent fashion. For the node BID, the curves estimated from the higher number of repetitions show the presence of a maximum around timestep 20. However, this maximum is not correctly estimated at lower number of repetitions. This example shows that an efficient simulator performing high number of repetitions

TABLE 2
Attractor Search Benchmark

Model	#nodes	Tool	#Start states	Time	Time per state	#attractors
T-LGL	51	BoolNet	2^{12}	0.12 s	32 us	3
		FPGA	2^{28}	6.46 s	0.024 us	5
CRPC	91	BoolNet	2^{16}	13.43 s	205 us	1
		FPGA	2^{20}	1.64 s	1.56 us	135
Fumia	98	BoolNet	2^{16}	15.62 s	238 us	4
		FPGA	2^{20}	0.59 s	0.56 us	26
CAC	70	BoolNet	2^{16}	20.17 s	307 us	4
		FPGA	2^{25}	1.59 s	0.047 us	6
FA-BRCA	28	BoolNet	2^{28}	214.8 s	0.8 us	1
		FPGA	2^{28}	4.31 s	0.016 us	1
MAPK	53	BoolNet	2^{16}	12.12 s	184 us	4
		FPGA	2^{24}	13.01 s	0.7 us	10

Summary of the evaluated models and results for the synchronous attractor search comparing BoolNet and our framework.

with low runtime helps to better capture the dynamics of a biological system.

4.2 Attractor Analysis

For attractor analysis, we excluded BooleanNet from the benchmark given its poor performance. BoolNet does not perform exhaustive attractor analysis if the number of nodes in the model is greater than 29, and all models considered in the benchmark exceed this limit. Hence, we selected a limited set of initial states and we have set the method for finding attractors to *chosen*, an option that allows us to guide the attractor analysis and to ensure a fair comparison with our framework.

Attractor search is performed by running multiple synchronous simulations using different initial states. This reduces the attractor search space only to simple attractors, but since we are interested in comparing runtime performances, it does not constitute a limitation for us. The time to generate the start states is excluded from timing measurements. The number of start states has been selected such that the runtime for a specific model is sufficiently long (>10 s) to avoid side effects for short runs (e.g.: effects of unexpected processes run by the operative system).

Table 2 summarizes the evaluated models and the measured runtime for both BoolNet and our accelerator framework. Since the FPGA simulations run consistently faster compared to BoolNet ones, the number of initial states has been adjusted accordingly. When we used the same number of start states for both, it often happened that either BoolNet ran for too long or the accelerator was too fast, resulting in short runtimes prone to side effects affecting performance measurements, e.g., time fluctuations due to other operative system processes running.

As the overall runtime is dependent on the number of start states used for the attractor search, Table 2 includes a runtime per state column to make the tools comparable. The speedup factors of the FPGA framework range from 50x to 6531x over BoolNet. Interestingly, the runtime of BoolNet per state is significantly better on the FA-BRCA model, the smallest model in terms of number of nodes, than on the other models. This is probably due to the fact that in this regime it can perform an exhaustive search of all states.

Performance Projections. As we run BoolNet only on a single core, a single accelerator core has been used on the FPGA to make the comparison fair. The accelerator consumes only 2 percent of the overall resources available on the FPGA. This allows the accelerator core to be replicated at least 20 times on a single FPGA card, resulting in a further speedup of 20x. The server system allows to plug in an additional coherent accelerator processor interface-based (CAPI-based) accelerator to further increase performance and adjust for a multi-threaded software implementation. When utilizing all cores of the POWER8 processor, the performance of the software should increase linearly and be 20 times faster as well.

TABLE 3
FPGA Resources Requirements

Model	LUTs	BRAM	LUTs (%)	BRAM (%)
T-LGL	7227	43	2.1	4.0
CRPC	7345	75	2.2	6.9
Fumia	7405	75	2.2	6.9
PSL	54945	281	16.5	26.0

Required FPGA resources for the core per model and the property specification language (PSL).

Comparison with Symbolic Approaches. We also ran boolSim [28] (previously known as *genYsis*, a symbolic ROBDD-based tool for attractor analysis on our models. Synchronous attractor computation for T-LGL (51 nodes) on an Intel Xeon processor running at 3.5GHz was performed in only 80 seconds and 71 attractors were reported. boolSim took 10 seconds to finish the analysis for MAPK (53 nodes). However, we observed that boolSim was unable to finish the attractor analysis in a reasonable time as the number of nodes increases. Some of the runs of boolSim had to be killed after running for a long time. For instance, for the Fumia model (98 nodes), boolSim kept running for >5317 minutes (approximately 4 days). The runtime for both CRPC (91 nodes) and CAC (70 nodes) is >8352 minutes (approximately 6 days). We chose to stop boolSim after running for such a long time. As it is observed in all symbolic approaches, no response/feedback was presented to the user during this time. Our approach, on the other hand, performs an exhaustive search and presents the results fast and efficiently thanks to its FPGA acceleration.

For the sake of completeness, we also tried to test the state-of-the-art ROBDD-based software tool *geneFAtt* [29]. Although the source code is publicly available, it seems to be incomplete, and compilation failed on the POWER system as well as on an x86 system.

4.3 Further Improvements

FPGA Utilization. The simulation core is rather small leaving the FPGA resources under utilized. Each model requires around 7,200 to 7,400 look-up tables (LUTs) which is about 2 percent of the overall resources available on the KU060 FPGA. The BlockRAM requirements are higher due to the collector module. The entire core requires 43 BlockRAM instances for the T-LGL model. Each of the larger models requires 75 instances, which is about 7 percent of all BlockRAMs on this FPGA chip. The POWER Service Layer and the interconnecting modules require far more logic resources and BlockRAMs. These are necessary to connect the accelerator to the host system. Table 3 summarizes the required resources.

Performance Enhancements. As the core requires little resources on the FPGA, multiple instances of the Boolean network can be analysed concurrently to further reduce the processing time. This will more efficiently utilize the available bandwidth towards the processor. More results can be concurrently sent back to the host system. Since results are sent back only after all simulation repetitions are complete, a single core requires a high bandwidth. The experiments indicate a utilization of less than 1percent of the available bandwidth of CAPI. Recent research has demonstrated the use of network-attached FPGAs to accelerate applications [50]. Boolean network simulations can leverage such an architecture by distributing the simulations across multiple FPGAs. This will allow to scale the models even further without sacrificing performance.

5 Discussion

In this work we have presented an FPGA-based framework for the simulation of Boolean models and the computation of attractors. We show that our accelerator can be used to asynchronously simulate network dynamics more efficiently than with the existing software solutions. The proposed framework exhibits an order of magnitude speedup over existing multi-threaded software tools. We also leverage the speedup offered by our accelerator to perform a massive number of repetitive asynchronous simulations of the T-LGL model. Our framework successfully computes simple attractors of large and complex Boolean models, exhibiting one to three orders of magnitude speedup over existing software solutions.

Our results show that our solution enables analysis of Boolean models with unmatched performance. The low utilization of the FPGA observed in the analyzed models, shows that there is enormous room for improvement in terms of speed. A straightforward way to achieve additional speedup is to synthesize multiple instances of the Boolean model on the FPGA. In addition, simulations can also be distributed across multiple FPGAs, if further speedup is necessary. Gaining speed eliminates existing limitations for Boolean model analysis in terms of number of nodes and model complexity that a simulator can handle. Being able to simulate and analyze larger and more complex Boolean networks, up to thousands of nodes, allows us to consider a more comprehensive description of a biological system and to fully exploit the potential of high-throughput molecular data.

Besides performance improvements, our framework can be easily extended to use other update strategies, such as random ranked updates. This will increase its ability to explore the state space, hence improving attractor detection. Additionally, another intriguing extension consists in implementing complex attractor computation on FPGAs to enable fast analysis of the reachable states of a Boolean model.

The integration of the accelerator with a POWER8 processor via CAPI greatly simplifies its usage. We believe that this is a fundamental feature in making our framework a valuable tool for the whole scientific community, offering the possibility to seamlessly integrate it in software applications.

Acknowledgments

The projects leading to this publication have received funding from the European Union's Horizon 2020 research and innovation programme under grant agreements No 668858 and No 826121.

References

[1] L. Koch, "A global view of regulatory networks," *Nature Rev. Genetics*, vol. 17, no. 5, pp. 252–252, Mar. 2016. [Online]. Available: http://www.nature.com/articles/nrg.2016.36

[2] L. Glass and S. A. Kauffman, "The Logical Analysis of Continuous, Non-linear Biochemical Control Networks," *J. Theoretical Biol.*, vol. 39, pp. 103–129, 1973.

[3] I. N. Melas, A. D. Chairakaki, E. I. Chatzopoulou, D. E. Messinis, T. Katopodi, V. Pliaka, S. Samara, A. Mitsos, Z. Dailiana, P. Kollia, and L. G. Alexopoulos, "Modeling of signaling pathways in chondrocytes based on phosphoproteomic and cytokine release data," *Osteoarthritis Cartilage*, vol. 22, no. 3, pp. 509–518, 2014.

[4] J. Lu, H. Zeng, Z. Liang, L. Chen, L. Zhang, H. Zhang, H. Liu, H. Jiang, B. Shen, M. Huang, et al., "Network modelling reveals the mechanism underlying colitis-associated colon cancer and identifies novel combinatorial anti-cancer targets," *Sci. Rep.*, vol. 5, 2015, Art. no. 14739.

[5] H. Chen, G. Wang, R. Simha, C. Du, and C. Zeng, "Boolean models of biological processes explain cascade-like behavior," *Sci. Rep.*, vol. 6, 2016, Art. no. 20067.

[6] Y. Hu, Y. Gu, H. Wang, Y. Huang, and Y. M. Zou, "Integrated network model provides new insights into castration-resistant prostate cancer," *Sci. Rep.*, vol. 5, no. April, pp. 1–12, Nov. 2015.

[7] H. F. Fumiã and M. L. Martins, "Boolean network model for cancer pathways: Predicting carcinogenesis and targeted therapy outcomes," *PLoS One*, vol. 8, no. 7, Sep. 2013, Art. no. 11.

[8] L. Grieco, L. Calzone, I. Bernard-Pierrot, F. Radvanyi, B. Kahn-Perlès, and D. Thieffry, "Integrative modelling of the influence of MAPK network on cancer cell fate cecision," *PLoS Comput. Biol.*, vol. 9, no. 10, pp. 1–15, Sep. 2013.

[9] D. P. Cohen, L. Martignetti, S. Robine, E. Barillot, A. Zinovyev, and L. Calzone, "Mathematical modelling of molecular pathways enabling tumour cell invasion and migration," *PLoS Comput. Biol.*, vol. 11, no. 11, Sep. 2015, Art. no. e1004571. [Online]. Available: http://dx.plos.org/10.1371/journal.pcbi.1004571

[10] J. Saez-Rodriguez, L. Simeoni, J. A. Lindquist, R. Hemenway, U. Bommhardt, B. Arndt, U. U. Haus, R. Weismantel, E. D. Gilles, S. Klamt, and B. Schraven, "A logical model provides insights into T cell receptor signaling," *PLoS Comput. Biol.*, vol. 3, no. 8, pp. 1580–1590, Sep. 2007. [Online]. Available: http://dx.plos.org/10.1371%2Fjournal.pcbi.0030163

[11] J. Dorier, I. Crespo, A. Niknejad, R. Liechti, M. Ebeling, and I. Xenarios, "Boolean regulatory network reconstruction using literature based knowledge with a genetic algorithm optimization method," *BMC Bioinf.*, vol. 17, no. 1, 2016, Art. no. 410. [Online]. Available: https://bmcbioinformatics.biomedcentral.com/articles/10.1186/s12859-016 -1287-z

[12] A. Garg, A. DiCara, I. Xenarios, L. Mendoza, and G. De Micheli, "Synchronous versus asynchronous modeling of gene regulatory networks," *Bioinf.*, vol. 24, no. 17, pp. 1917–1925, 2008.

[13] S. Kauffman, "Homeostasis and differentiation in random genetic control networks," *Nature*, vol. 224, no. 5215, pp. 177–178, Oct. 1969. [Online]. Available: http://dx.doi.org/10.1038/224177a0

[14] R. Thomas, "Regulatory networks seen as asynchronous automata: A logical description," *J. Theoretical Biol.*, vol. 153, no. 1, pp. 1–23, 1991. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0022519305803509

[15] M. Purandare, R. Polig, and C. Hagleitner, "Accelerated analysis of Boolean gene regulatory networks," in *Proc. 27th Int. Conf. Field Programmable Logic Appl.*, 2017, pp. 1–6.

[16] A. Fauré, A. Naldi, C. Chaouiya, and D. Thieffry, "Dynamical analysis of a generic Boolean model for the control of the mammalian cell cycle," *Bioinf.*, vol. 22, pp. 124–131, 2006.

[17] F. Li, T. Long, Y. Lu, Q. Ouyang, and C. Tang, "The yeast cell-cycle network is robustly designed," *Proc. Nat. Acad. Sci.*, vol. 101, no. 14, pp. 4781–4786, 2004. [Online]. Available: http://www.pnas.org/cgi/doi/10.1073/pnas.0305937101

[18] S. Huang, "Gene expression profiling, genetic networks, and cellular states: An integrating concept for tumorigenesis and drug discovery," *J. Molecular Med.*, vol. 77, no. 6, pp. 469–480, 1999.

[19] D. A. Orlando, C. Y. Lin, A. Bernard, J. Y. Wang, J. E. Socolar, E. S. Iversen, A. J. Hartemink, and S. B. Haase, "Global control of cell-cycle transcription by coupled CDK and network oscillators," *Nature*, vol. 453, no. 7197, pp. 944–947, 2008.

[20] T. Skodawessely and K. Klemm, "Finding attractors in asynchronous Boolean dynamics," *Adv. Complex Syst.*, vol. 14, no. 3, pp. 439–449, 2010. [Online]. Available: http://arxiv.org/abs/1008.3851

[21] T. Akutsu, S. Kuhara, O. Maruyama, and S. Miyano, "A system for identifying genetic networks from gene expression patterns produced by gene disruptions and overexpressions," *Genome Informat.*, vol. 9, pp. 151–160, 1998.

[22] S. Q. Zhang, M. Hayashida, T. Akutsu, W. K. Ching, and M. K. Ng, "Algorithms for finding small attractors in boolean networks," *Eurasip J. Bioinf. Syst. Biol.*, vol. 2007, no. 1, 2007, Art. no. 20180.

[23] C. Müssel, M. Hopfensitz, and H. A. Kestler, "BoolNet-an R package for generation, reconstruction and analysis of Boolean networks," *Bioinf.*, vol. 26, no. 10, pp. 1378–1380, 2010.

[24] R. Zhang, M. V. Shah, J. Yang, S. B. Nyland, X. Liu, J. K. Yun, R. Albert, and T. P. Loughran, "Network model of survival signaling in large granular lymphocyte leukemia," *Proc. Nat. Acad. Sci.*, vol. 105, no. 42, pp. 16 308–16 313, 2008. [Online]. Available: http://www.pnas.org/cgi/doi/10.1073/pnas.0806447105

[25] A. Rodríguez, D. Sosa, L. Torres, B. Molina, S. Frías, and L. Mendoza, "A Boolean network model of the FA/BRCA pathway," *Bioinf.*, vol. 28, no. 6, pp. 858–866, 2012.

[26] I. Albert, J. Thakar, S. Li, R. Zhang, and R. Albert, "Boolean network simulations for life scientists," *Source Code Biol. Med.*, vol. 3, no. 1, 2008, Art. no. 16.

[27] N. Miskov-Zivanov, A. Bresticker, D. Krishnaswamy, S. Venkatakrishnan, D. Marculescu, and J. R. Faeder, "Emulation of biological networks in reconfigurable hardware," in *Proc. 2nd ACM Conf. Bioinf. Comput. Biol. Biomed.*, 2011, pp. 536–540. [Online]. Available: http://dl.acm.org/citation.cfm?id=2147805.2147893

[28] A. Garg, A. Di Cara, I. Xenarios, L. Mendoza, and G. De Micheli, "Synchronous versus asynchronous modeling of gene regulatory networks," *Bioinf.*, vol. 24, no. 17, pp. 1917–1925, 2008. [Online]. Available: http://dx.doi.org/10.1093/bioinformatics/btn336

[29] D. Zheng, G. Yang, X. Li, Z. Wang, F. Liu, and L. He, "An efficient algorithm for computing attractors of synchronous and asynchronous boolean networks," *PLoS One*, vol. 8, no. 4, pp. 1–7, Sep. 2013. [Online]. Available: https://dx.plos.org/10.1371/journal.pone.0060593

[30] L. Salwinski and D. Eisenberg, "In silico simulation of biological network dynamics," *Nature Biotechnology*, vol. 22, no. 8, pp. 1017–1019, Aug. 2004. [Online]. Available: http://dx.doi.org/10.1038/nbt991

[31] G. Stoll, B. Caron, E. Viara, A. Dugourd, A. Zinovyev, A. Naldi, G. Kroemer, E. Barillot, and L. Calzone, " MaBoSS 2.0: An environment for stochastic Boolean modeling," *Bioinf.*, vol. 33, no. 14, pp. 2226–2228, Jul. 2017. [Online]. Available: https://academic.oup.com/bioinformatics/article/33/14/2226/3059141

[32] N. Berntenis and M. Ebeling, "Detection of attractors of large Boolean networks via exhaustive enumeration of appropriate subspaces of the state space," *BMC Bioinf.*, vol. 14, no. 1, 2013, Art. no. 361.

[33] A. Saadatpour, R. Albert, and T. C. Reluga, "A reduction method for Boolean network models proven to conserve attractors," *SIAM J. Appl. Dynamical Syst.*, vol. 12, no. 4, pp. 1997–2011, 2013. [Online]. Available: http://epubs.siam.org/doi/10.1137/13090537X

[34] F. Ay, F. Xu, and T. Kahveci, "Scalable steady state analysis of boolean biological regulatory networks," *PLoS One*, vol. 4, no. 12, pp. 1–9, Dec. 2009.

[35] N. D. Mendes, R. Henriques, E. Remy, J. Carneiro, P. T. Monteiro, and C. Chaouiya, "Estimating attractor reachability in asynchronous logical models," *Frontiers Physiology*, vol. 9, 2018, Art. no. 1161.

[36] D. Zheng, G. Yang, X. Li, Z. Wang, F. Liu, and L. He, "An efficient algorithm for computing attractors of synchronous and asynchronous boolean networks," *PLoS One*, vol. 8, no. 4, pp. 1–7, Apr. 2013.

[37] A. Mizera, J. Pang, H. Qu, and Q. Yuan, "Taming asynchrony for attractor detection in large boolean networks," *IEEE/ACM Trans. Comput. Biol. Bioinf.*, vol. 16, no. 1, pp. 31–42, Jan./Feb. 2019.

[38] E. Dubrova and M. Teslenko, "A SAT-based algorithm for finding attractors in synchronous boolean networks," *IEEE/ACM Trans. Comput. Biol. Bioinf.*, vol. 8, no. 5, pp. 1393–1399, Sep. 2011.

[39] W. Guo, G. Yang, W. Wu, L. He, and M. Sun, "A parallel attractor finding algorithm based on boolean satisfiability for genetic regulatory networks," *PLoS One*, vol. 9, no. 4, pp. 1–10, Sep. 2014.

[40] G. Arellano, J. Argil, E. Azpeitia, M. Benítez, M. Carrillo, P. Góngora, D. A. Rosenblueth, and E. R. Alvarez-Buylla, ""Antelope": A hybrid-logic model checker for branching-time Boolean GRN analysis," *BMC Bioinf.*, vol. 12, no. 1, 2011, Art. no. 490.

[41] I. Tagkopoulos, C. Zukowski, G. Cavelier, and D. Anastassiou, "A custom fpga for the simulation of gene regulatory networks," in *Proc. 13th ACM Great Lakes Symp. VLSI*, 2003, pp. 132–135.

[42] M. Zerarka, J. David, and E. M. Aboulhamid, "High speed emulation of gene regulatory networks using FPGAs," in *Proc. 47th Midwest Symp. Circuits Syst.*, Aug. 2004, pp. I–545.

[43] I. Pournara, C. Bouganis, and G. Constantinides, "FPGA-accelerated Bayesian learning for reconstruction of gene regulatory networks," in *Proc. Int. Conf. Field Programmable Logic Appl.*, Sep. 2005, pp. 323–328.

[44] R. Ferreira and J. C. Goldner Vendramini, "FPGA-accelerated attractor computation of scale free gene regulatory networks," in *Proc. Int. Conf. Field Program. Logic Appl.*, Aug. 2010, pp. 550–555.

[45] J. F. Keane, C. Bradley, and C. Ebeling, "A compiled accelerator for biological cell signaling simulations," in *Proc. ACM/SIGDA 12th Int. Symp. Field Program. Gate Arrays*, 2004, Art. no. 233. [Online]. Available: http://portal.acm.org/citation.cfm?doid=968280.968313

[46] N. Miskov-Zivanov, A. Bresticker, D. Krishnaswamy, S. Venkatakrishnan, P. Kashinkunti, D. Marculescu, and J. R. Faeder, "Regulatory network analysis acceleration with reconfigurable hardware," in *Proc. Annu. Int. Conf. IEEE Eng. Med. Biol. Soc.*, vol. 2011, Aug. 2011, pp. 149–152. [Online]. Available: http://ieeexplore.ieee.org/document/6089916/%5Cn, http://www.ncbi.nlm.nih.gov/pubmed/22254272

[47] L. B. da Silva, D. Almeida, J. A. M. Nacif, I. Sánchez-Osorio, C. A. Hernández-Martínez, and R. Ferreira, "Exploring the dynamics of large-scale gene regulatory networks using hardware acceleration on a heterogeneous cpu-fpga platform," in *Proc. Int. Conf. ReConFigurable Comput. FPGAs*, Dec. 2017, pp. 1–7.

[48] C. Chaouiya, D. Bérenguier, S. M. Keating, A. Naldi, M. P. van Iersel, N. Rodriguez, A. Dräger, F. Büchel, T. Cokelaer, B. Kowal, B. Wicks, E. Gonçalves, J. Dorier, M. Page, P. T. Monteiro, A. von Kamp, I. Xenarios, H. de Jong, M. Hucka, S. Klamt, D. Thieffry, N. Le Novère, J. Saez-Rodriguez, and T. Helikar, "Sbml qualitative models: a model representation format and infrastructure to foster interactions between qualitative modelling formalisms and tools," *BMC Syst. Biol.*, vol. 7, no. 1, Dec. 2013, Art. no. 135.

[49] A. Saadatpour, R. S. Wang, A. Liao, X. Liu, T. P. Loughran, I. Albert, and R. Albert, "Dynamical and structural analysis of a T cell survival network identifies novel candidate therapeutic targets for large granular lymphocyte leukemia," *PLoS Comput. Biol.*, vol. 7, no. 11, Sep. 2011, Art. no. e1002267. [Online]. Available: http://dx.plos.org/10.1371/journal.pcbi.1002267

[50] J. Weerasinghe, R. Polig, F. Abel, and C. Hagleitner, "Network-attached fpgas for data center applications," in *Proc. IEEE Int. Conf. Field-Programmable Technol.*, 2016, pp. 36–43.