Reconfigurable FPGA-Based Acceleration of RTL Simulation and Co-Simulation for Cloud Datacenter Workloads

Kaushik Velapa Reddy
Microsoft Corporation, United States of America

ABSTRACT

The complexities that have arisen in the form of increasing workloads in cloud datacenters are further complicating the efficient Register Transfer Level (RTL) simulation and hardware/software co-simulation of high-performance computing platform design and verification. Traditional software-based RTL simulation can experience long response times and high computation time and are limited in scalability when performing an analysis of processor, accelerator, memory, and interconnect architecture on realistic workloads in datacenters. The paper provides a scaling FPGA-based accelerator model to leverage the performance of RTL simulation and co-simulation, through the utilization of hardware to run computationally intensive aspects of the simulation. The suggested model dynamically deploys the chosen RTL simulation kernels to FPGA resources as well as keeps in sync with a software simulation environment. A configurable communication and synchronization interface enables communication between host based simulators and FPGA accelerated modules and enables flexible workload and rapid architecture exploration. The framework is suited in the case of cloud datacenter with heterogeneous processors, accelerators, hierarchies in memory as well as high speed interconnection. Performance with respect to the simulation execution time, ratio of acceleration, and utilization of FPGA resources, overhead of synchronization and scalability across workload settings are measured. The gradually increasing reduction of simulation latency with modeling accuracy and rapid validation of hardware-software design can be approached using the suggested approach that provides a scalable foundation.

Keywords: FPGA Acceleration, RTL Simulation, Hardware–Software Co-Simulation, Cloud Datacenters, Reconfigurable Computing, Hardware Simulation, Workload Modeling.

DOI: 10.21590/ijhit.05.01.8

I. INTRODUCTION

The fast pace of cloud computing usage, as well as large-scale datacenter infrastructures has added much complexity to the modern computing systems. The design of modern datacenter platforms features heterogeneous processors, accelerators, memory subsystems, high-speed interconnects, storage, and specialized hardware components to support the needs of the intensive workload, such as artificial intelligence, data analytics, virtualization, and distributed services. Such systems have to be designed and validated by precise modeling of the hardware behaviour prior to physical implementation. A well-known method to verify the functionality of digital hardware, timing behavior and communication, and architecture interactions is register-transfer level (RTL) simulation. The increasing mass (and complexity) of current hardware implementations, however, have made conventional software-based RTL simulation all the more time-consuming and computationally expensive. Overhead of the simulation can be a significant limitation in a situation where one must search through big populations of workloads and architecture.

Traditionally software RTL simulators run hardware events sequentially or with limited parallelism on general-purpose processors. These methods, despite their handy behavioural visibility and robust verification properties, may consume up to a large amount of time to run with an increase in the design complexity and simulation length. This problem scales up to cloud datacenter workloads, where system-level research may require re-simulation of an assortment of oscillating heterogeneous components, massive communication, memory transactions and workload-sensitive hardware behavior. It has hence been a research trend to accelerate the process of simulation of the RTL with not much loss of the accuracy. Another possible option would be the FPGA-based acceleration as FPGAs can have customizable hardware capabilities, and this can be utilized to execute the chosen simulation operations simultaneously.

Reconfigurable hardware can also potentially minimize the amount of time required to simulate, but can be made to allow the hardware architecture to be reconfigured to other hardware models and workload characteristics.

System simulation in FPGA has shown that it is possible to run cycle-accurate models on performance more than an order of magnitude higher than software-only techniques. An example is the use of FPGA platforms as a representative model to provide cycle-precise scale-out simulation of computer systems and can be employed to study large-scale systems in cloud-based environments (see fig. 1). These architectures demonstrate how the reconfigurable hardware with software infrastructure can be employed to enable scalable simulation of complex systems. This is what causes this approach to be cloud-oriented of particular concern to the datacenter workload analysis as it enables hardware models to be sold as scalable simulation resources, rather than being locked to local simulation environments. But the acceleration efficiency of FPGA depends on the optimal mapping of simulation workloads, seamless interaction between the hardware and the software components and synchronization control systems.

It is also necessary to accelerate since software RTL simulation is getting more and more costly to compute. Studies of RTL simulation acceleration have found that software-based simulation can prove to be a major limitation as verification (or circuit size) cost increases [2]. Software simulators are flexible and have mature mechanisms to do verification, but the model of execution does not necessarily use fine-grained parallelism that may exist in hardware descriptions. This allows an opportunity to delegate the computationally intensive simulations to special acceleration graphs and maintain the use of software components to regulate, set up, and test. This mix strategy can therefore have a compromise between the flexibility of the simulation and the effectiveness of hardware execution.

Hardware design and simulation environment also demonstrate the importance of having one flow through the design, simulation and implementation. An integrated SoC development system would connect the hardware construction, simulation and implementation into a single infrastructure [3]. This type of integration applies to an FPGA-based RTL acceleration system because the acceleration simulation environment must be designed to support a broad selection of hardware choices and the modeling of the desired hardware should be identical to the modeling of the intended hardware. It is more vital at cloud datacenter workloads as the systems that it is constructed on may contain unmatched processing units and programmable hardware components.

Hardware-software co-design is another precious hint to the breakthrough of limitations related to traditional RTL simulation. Simulation control and other activities of the host processor could be matched and run with a set of simple hardware accelerators, each with the designated portion of the computation. By doing so, the simulation capability can be separated according to the dynamics of single tasks and can be effective in terms of upgrading the overall performance [4]. However, movement of data, synchronization and events order, as well as communication overhead, should be well handled to achieve successful hardware-software partitioning. Unless these factors are controlled it is possible that the advantages of hardware acceleration are minimized by too much interaction between the FPGA and the host system.

Parallel RTL simulation techniques is another mechanism that enables simulation to be scaled up. The partitioning according to replication can take advantage of parallelism in RTL simulation and raise significantly simulation throughput of appropriate designs [5]. These strategies suggest that the speed of simulation may not solely be influenced by the optimization method but instead the simulation can be benefited by parallelism, parts calculations, dedicated hardware execution, and effective synchronization.

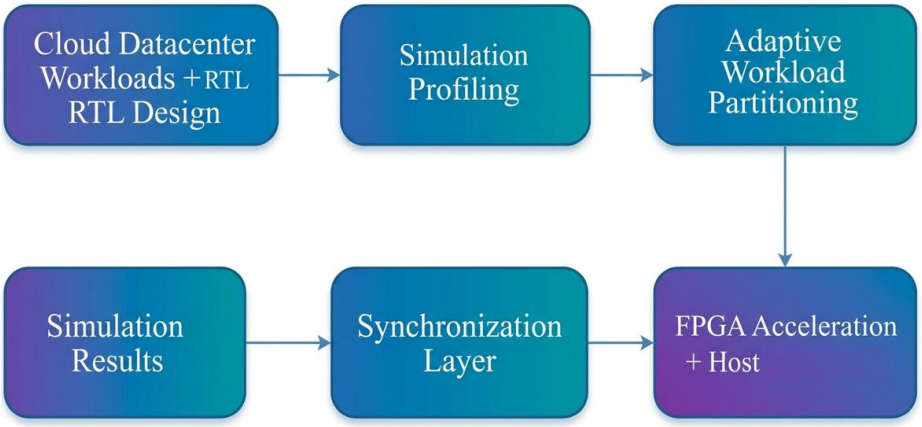


Figure 1- Proposed reconfigurable FPGA–host framework for accelerated RTL simulation and co-simulation of heterogeneous cloud datacenter workloads.

This notwithstanding, a cohesive reconfigurable FPGA-based platform that is specifically orientated towards speeding up RTL simulation and hardware-software co-simulation to cloud datacenter workloads is required. This framework needs to be dynamically capable of deciding the elements of a computationally intensive simulation and map the right operations to FPGA elements, schedule the execution between the FPGA and the host processor, and permit workload-adaptive reconfigurability. The other performance aspects that it will have to consider is the performance of simulation execution time, acceleration ratio, FPGA resource allocation, synchronization overhead and scalability. Achieving these demands could provide a flexible foundation to which to simulate more easily the increasingly more complicated datacenter hardware designs and yet at the same time provide the configurability of the different RTL implementations and cloud workload states.

II. RELATED WORK

Research on accelerating hardware and system simulation has bifurcated towards traditional software-based simulation to parallel, GPU-based, FPGA-based, as well as co-design of hardware and software. The increasing complexity of modern processors, system-on-chip (SoC) organizations, accelerators, and multi-processor clouds is driving this. Contributions made in the literature on the minimization of simulation time, scalability and use of special hardware platform with the use of special hardware platform find their ways in minimization of a single reconfigurable FPGA-based system but on the other hand it reveals the potential solutions to simulation and co-simulation of RTL with cloud-datacenter applications.

Karandikar et al. introduce FireSim which is an FPGA based and cycle-exact simulation platform [1] that is used to simulate a scale-out system on the public cloud. The work proved that the implementation FPGA-based could be used to model a detailed hardware, and also generate a scalable infrastructure to simulate large-scale computer-systems. FireSim particularly can be applied to the concept of cloud research, in which the cycle-accurate hardware modeling is indirectly connected to the distributed simulation resources. It has an architecture with the potential to provide simulation of a system using reconfigurable hardware that is no longer limited to the performance limits of software execution. Nevertheless, its main focus lies more in the system-level level simulation infrastructure than a generalized system that allows the dynamically partitioning of RTL simulation and co-simulation loads amongst FPGA and host software units.

Beamer [2] investigated the software based RTL simulation performance limitations and developed the motivation of dedicated acceleration. This paper highlighted the complexity of calculating the execution of the description of an operating system in RTL descriptions in more generalized processors and recommended more usage of hardware resources to accelerate simulation. The work offers a valuable conceptual basis to FPGA-based RTL acceleration since it is found that performance of software simulation is a bottleneck on hardware verification and architectural exploration. However, the study simply establishes the foundation in favor of acceleration being implemented beyond

the scale of what it provides of a complete scale cloud-based reconfigurable architecture that might be utilized to align the resources within the simulation and the workload of the heterogeneous datacenters.

Amid et al. [3] developed a mixed hardware-software platform named Chipyard that was used to design, simulate and deploy custom SoCs. The framework provides a shared ground between hardware generation and simulation and implementation processes, and reduces the degree of discontinuousness of different phases of hardware development. Its integration based approach can also be applied to high-speed RTL simulation as an FPGA based simulator needs the consistency of the results of the RTL design, simulation model and hardware realization. One more thing that Chipyard has taught me is the importance of a flexible infrastructure in the experimentation of different architectural designs. However, its main goal is to generalize SoC design and development as opposed to streamlining the usage of the FPGA resources to achieve RTL-simulation and hardware-software co-simulation in cloud datacenter settings.

Elsabbagh et al. [4] studied hardwaresoftware co-design as a technique of increasing the speed of RTL simulation. They keep the simulation tasks software-assisted and proper hardware to enable computing intensive tasks to be run more efficiently without sacrificing the freedom and control of software. This is particularly noteworthy to FPGA-based simulation as holistic hardware integration may not even be practical with an ambitious RTL model. An effectively designed partitioning mechanism may be employed to assign aiming simulation kernels that should be done on FPGA and outsource control-sensitive or less parallel tasks to the host core processor. The paper has proven that co-design may improve the performance of the simulation, but the correspondence of the target environment to the heterogeneous workloads of the cloud datacenters in terms of partitioning, communication, and synchronization as well as the reconfiguration can also be discussed.

Wang and Beamer [5] proposed RepCut which is a parallel RTL simulation model, which is based on partitions replication. Their paper shows that a keen exploitation of the structure of RTL designs can give significant parallel simulation benefits. This replication scheme allows simulation work loads to be distributed such that can deliver superlinear performance enhancement under the appropriate circumstances. The importance of this study is that it demonstrates that simulation acceleration is not solely accomplished merely by the presence of faster execution hardware, but of decent workload decomposition. With an acceleration based on FPGA, independent or replicated simulation regions and which can be executed concurrently can be determined by the principles of partitioning. But RepCut is easily parallelizable to RTL simulation and provides no support to a complete FPGA host co-simulation architecture to reach cloud-scale workload conditions.

Zhou et al. [6] proposed Khronos hardware RTL simulation that aims at improving hardware RTL simulation through optimization of memory access. The article is an analysis of the memory behavior as a vital parameter that influences the performance of simulation and it demonstrates that when the undesired memory activities are reduced; simulation performance would be improved. This implementation mode complies with the FPGA acceleration because the FPGA simulation architecture may also possess memory-bandwidth, and data-transfer limitations. This can be the case in high utilization in the accelerator, as a quick access to memory is required. It offers a beneficial optimization perspective, although it is largely about memory-access behavior, rather than a reconfigurable FPGA model integrating simulation compute, host execution and cloud workload management.

The authors of [7] Lopez-Paradis et al. were concerned with the fast computing of very large SoC design using Metro-MPI. Their work shows the need to implement execution distributedly where the RTL models are too many cycles to be simulated in a normal simulation environment. Message passing is used, which offers a product distribution mechanism of distributing simulation workloads over processing resources. The cloud datacenter simulation can be subject to this strategy as cloud systems naturally constitute distributed computing resources. However, the propagation of software deployment can introduce additional weight to communications and coordination, an opportunity to switch to use specific FPGA acceleration to aspects of the simulation that have significant computational burdens and execute them closer to the machine.

Lin et al. [8] researched an implementation of a RTL simulation flow on a GPU that transforms RTL processing into CUDA processing and uses batch stimulus to utilize the parallelism in the GPUs. Through their work, it is

demonstrated that massively parallel processing architectures can be used to efficiently run RTL workloads faster. Execution synchronized Batch execution may augment utilization in the hardware where a number of regular simulation activities exist. Although the idea of GPUs promises a lot of computational throughput, the model of their execution differs in several aspects in comparison to FPGA-based acceleration; the configurability of the hardware, its hardware specialization, and mapping of resources. The work thus confirms the role played by GPU acceleration as a significant alternative besides encouraging research on the reconfigurable FPGA architectures towards application of customized hardware execution where the workload is helpful.

Simulation at the gate level studied by Chatterjee et al. [9] involves the use of the computerized core, a process that is also known as the usage of a totally computational model. The paper has demonstrated that graphics processors can be used to perform computational intensive simulation tasks, with fine grained parallelism. The work is a previous contribution towards dedicated hardware acceleration to support circuit simulation and is consistent with the general finding that simulating programs that run on the more usual CPU can actually be improved with parallel architectures. Nevertheless, computation implementation can often be very rigid to a relatively fixed computational model with the use of GPUs, but with the use of FPGA implementation, the hardware structures can be tailored to the characteristics of the simulation workload. This is an important distinction when there are heterogeneous RTL models and dynamically-evolving cloud datacenter workloads.

Babb et al. [10] researched on logic emulation with virtual wires, who provided one of the early examples of how configurable hardware resources can be utilized to accelerate hardware implementation and to eliminate communication bottlenecks. The work has established concepts that can be applied to the hardware based emulation including partitioning, cross-configurable resource communication. The principles may be extended to the simulation systems on the FPGA in the present day as they are able to correctly interconnect and coordinate to map the hardware models. However the assumptions in technology and systems of the early logic-emulation systems differ greatly with simulating environments in the cloud where logical scale is needed.

Ahmad et al. [11] investigated scalable parallel systems of events simulation on HDL multicore simulating systems. They do this by using multiple cores of processing capacity to more effectively simulate events. This is of specific interest to at least event parallelism as far as RTL simulation is concerned, in that large designs have many simulation events and their execution can be done in parallel. The importance of parallel execution and workload distribution on the work is imposed and memory-access, event-dependency and synchronization overheads overheads can cripple CPU-centric parallelism. An additional possibility would be FPGA acceleration, providing parallel execution architectures that are customizable.

Qian and Deng [12] discussed the idea behind accelerating the RTL simulation with the GAP and demonstrated how one can take an advantage of using data-level parallelism of a RTL workload. This acceptability of GPUs was also a result of their product which contributed to minimization of run time of simulations. Nevertheless, whether GPU acceleration is effective or not, is in the nature of the aptness of RTL operations to the GPU programming model. The solutions with FPGA provide greater extensibility in creating custom datapaths and control structures and thus are attractive to heterogeneous simulation kernels.

Nanjundappa et al. [13] developed a SystemC simulator called SCGPSim, which is a simulator chip (GPU) simulator providing better performance to the simulation. The book explains how it is possible to deployed modeling environments at system level and adapted to parallel custom processors. Simulation SystemC is significant in architectural exploration since it is capable of simulating interactions at higher levels than RTL, but in hardware-oriented ways. SystemC Acceleration There is no explicit support of reconfiguring FPGA in the form of GPU-based acceleration, or combined FPGA-software RTL simulation, but it remains open to a choice of using configurable hardware with conventional simulation environments.

Miller et al. [14] proposed a distributed parallel simulator, Graphite, which entails multicore architecture. According to the experiment, the distributed resource is applied in the architectural simulation and the process of fastening as well as the importance of scalability of the infrastructure simulating the building. Such distributed techniques can be applied to

the workloads in a cloud datacenter when it can dynamically schedule huge computing resources. But overhead can be introduced by communication and synchronization of distributed software processes. Accelerated with FPGA Distributed simulation can be enhanced with special hardware acceleration moving computational intensive tasks onto special hardware resources.

The idea of SlackSim, presented by Chen et al. was a parallel simulation front-end of chip multiprocessors on multiprocessor systems [15]. They investigated parallel simulation as a way of enhancing scalability of architectural studies in their work. The method proves that simulation as such can be enhanced using parallel hardware instead of just modelling parallel target systems. It can be generalised to the datacenter workload simulation in the cloud where a range of different architecture models and workload conditions may need to be tested. The parallel simulation using CPUs, however, has less configurability, and hardware specialization that may be provided by FPGA resources.

Finally, Tan et al. [16] suggested DIABLO which is a computer network simulator of warehouses based on FPGA. The work particularly applies to the analysis of the cloud datacenters as it explains the efficiency of using FPGA resources to model the behaviour of big computer-network. DIABLO shows that it can be used to run portions of warehouse-scale simulation, using specialized hardware and at the size of datacenter environments. But it deals mostly with network simulation as opposed to generalized RTL simulation and hardware software co-simulation.

III. RECONFIGURABLE FPGA–SOFTWARE CO-SIMULATION ARCHITECTURE FOR CLOUD WORKLOAD ACCELERATION

The architecture proposal introduces on board a reconfigurable FPGA-software co-simulation architecture that is used to accelerate the RTL simulation of the heterogeneous cloud datacenter workloads. The architecture is developed to rely on the fact that all simulation activities do not need an identical platform of execution. The most intensive and parallel based RTL functions are capable of being scheduled into FPGA resources, it is possible to do a whole scheduling of the most computationally and parallel intensive functions, but control, configuration and activity sensitive functions may be left on the host processor. It is a heterogeneous implementation model, that has the flexibility of software simulation and the parallel execution and flexibility of FPGA hardware.

The architecture is made up of six key functional layers, workload and RTL input management, simulation profiling, workload partitioning, FPGA acceleration, host-side simulation control, and synchronization and result management. As a coherent pipeline, these layers are configured as units, rather than discrete. The overall aim is to attain reduced time spent in simulation running and resemblance of the behavior of the software reference model and the FPGA-based speeded-up simulation route.

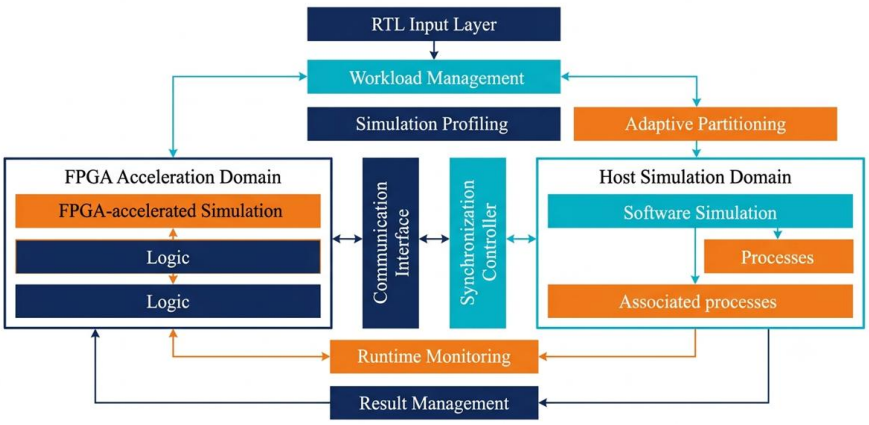


Figure 2- System architecture of the proposed reconfigurable FPGA–software co-simulation framework

3.1 Cloud Workload and RTL Input Layer

The first layer is provided with the RTL design, simulation setup, simulation test stimuli, and cloud datacenter workload. The processing units, memory controllers, accelerators, interconnects or other hardware units employed in the target architecture of the datacenter can also be the RTL input. The descriptions of workloads provide us with information about the trends of transactions, request of memory and events of communication among other activities that rely on workloads.

The input layer is preprocessed to convert the design and workload data given to a form that can be scanned to be simulated. The RTL modules can be examined based on their functional behavior, event activity and computational intensity, data dependency and communication needs. This representation provides the information which the further processes of profiling and partitioning require.

The framework will be configured to compliant heterogeneous datacenter environment in which diverse hardware will demonstrate a diversity simulation response. The speed of accelerator datapath can be significantly larger than a control component, as an example. When using both parts simultaneously, this could result in an inefficient use of simulation resources. Therefore, the structure considers the work load characteristics to make consequent mapping decisions.

3.2 RTL Simulation Profiling and Bottleneck Identification

Simulation profiling is done by the second layer to locate computationally interesting portions of the RTL model. The properties of frequency of events, intensity of execution, behavior of memory access, data dependencies and communication needs are analyzed through profiling. It is aimed at determining the possible simulation domains that can be good candidates of FPGA acceleration.

The profiling stage subdivides the simulation activities into coarse three categories i.e. FPGA appropriate computation, host oriented computation and host coordination functions. Operations typical of FPGA applications tend to be very parallel, predictable data flows, reuse of computation, or high frequency. Host based operations include simulation, configuration processing and irregular operation and inefficient functions, which are not realizing efficient implementation as dedicated hardware. The synchronization of state exchange between the FPga and the host, consistency, a communication, and coordination are part of coordination operations.

Random mapping of hardware is not made possible with this profiling mechanism. Instead of making the whole simulator of the RTL itself run on the FPGA, it is only those areas that are likely to benefit of being implemented on the hardware of choice. This is significant since there is a limit to the amount of FPGA resources that can be utilized, and over-hardware mapping can chew up less and less resources without a correspondingly accelerated simulation.

3.3 Adaptive RTL Workload Partitioning

The next step of profiling is the workload partitioning layer which assigns the simulation model to FPGA and the host processor. The dividing mechanism takes into consideration computational intensity, parallelism, data dependencies, frequency of communications, and availability of FPGA resources.

The partitioning introduces two synchronized executions domains. FPGA domain is where simulation kernels can be hardware accelerated and the other portion is where the host domain has simulation functions that are written in software. A very lightweight coordination layer provides an interface between the two domains, and synchronises the look and feel of transferring simulation states and input stimuli, intermediate values, and generated events.

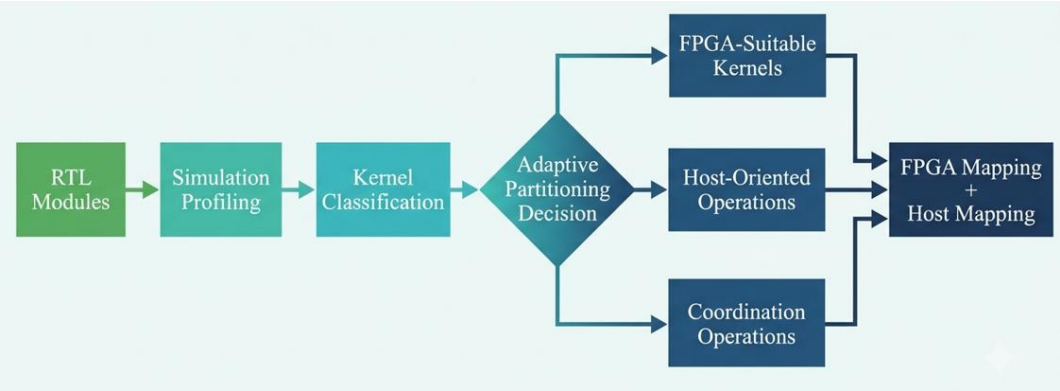


Figure 3- Adaptive RTL workload partitioning process for classifying simulation kernels

The partitioning approach is dynamic as RTL designs and cloud loads can yield varying acceleration opportunities. The portion of its simulation load could be allocated, with a larger proportion, to the FPGA, and less to the host, some analysis of configuration might have been performed. This flexibility enables the framework to deal with the Mayo of different datacenter designs without necessarily being forced to simulate each design with an alternate system simulator.

Amongst them is the reduction of over communication between the two domains. When a faster kernel requires executing state transfers with high frequency to a host, the latency benefits of FPGA execution can be offset by the overhead of communications. The partitioning layer will, therefore, aim at keeping the number of computational units in the FPGA domain as large as possible, but without paying too much attention to unneeded synchronization locations.

3.4 Reconfigurable FPGA Acceleration Layer

The primary hardware execution ability of the framework is FPGA acceleration layer. The selected RTL simulation kernels are optimized into hardware formats that can be programmed to the FPGA, and are overlaid onto configurable logic resources. Multiple parallel datapaths and various process structures can also be carried out using the FPGA to accommodate a simulation operation based on the workload target.

The architecture proposed focuses on reconfigurability. The FPGA resources are capable of being programmed to match the selected RTL kernels and workload properties rather than simply depending on the pre-determined accelerator to execute a single simulation model. Different workloads will be supported on different hardware accordingly and this is known as cloud workload. This provides a system of variation to accelerator resource to different simulation requirements.

The FPGA layer also includes local buffering and data-management schemes to reduce the data transfers that are unnecessary. Simulation data requiring high frequency can be co-located with the accelerated computation, with the bigger or less common state data existing in host memory. The mentioned organization facilitates the balanced use of resources, memory capacity and communication overhead in FPGA.

Accelerator produces simulation events, state updates and the information to host simulator is required. These findings are moved by the communication interface based on the synchronization requirements defined by co-simulation controller.

3.5 Host-Side Simulation and Control Layer

The host processor still has to be tasked with activities that have a software flexibility factor to it. They include simulator start, configuration control, job scheduling, executing the reference models, event coordination, logging, results collection and runtime monitoring.

It is also the host simulator which maintains the general state simulation and the reference in accordance with which the accelerated operations can be synchronized. When a simulation step that was running on a FPGA had reached its end,

its results would be given to the host machine where it may be further verified or processed. On the other hand, when needed, new inputs and state information are provided to the FPGA by the host.

All software-simulation capabilities in hardware do not have to be re-implemented by this division. It also enables the current software simulation infrastructure to still remain a part of the framework as comparisons of chosen computational bottlenecks are accelerated. So, the solution provided is centered on the part of the joint execution rather than the part of the software simulation replacement.

3.6 Hardware–Software Synchronization and Co-Simulation Control

The point of synchronization is an imperative component of the system as FPGA and host processor simulate different aspects of the same simulation. The synchronization layer matches the time of simulation, a sequence of events, change of states and data exchange.

Both directions of a point where the simulation is synchronized, the host and the FPGA exchange only the information necessary to maintain a steady state of the actual simulation. The framework is thus aimed at preventing redundant tie-ups following each single operation. Instead, the granularity of synchronization can be varied according to the workload dependencies, and the character of its implementation.

The order is determined by the co-simulation controller. The process is just an ordinary cycle where the host generates or captures simulation stimuli, and subsequently sends the appropriate inputs to the FPGA. The FPGA is then used to execute the selected simulation kernels in parallel and produce new states or events. These are relayed back to the host where the processes which are software-dependent are applied. This is carried on until the state of completion of the simulation work load, is reached.

Synchronization overhead is explicitly considered as performance factor. The ratio of high acceleration can be used only when the cost of communication and coordination is low enough. Therefore, the framework takes into account the computation time of FPGA as well as the costs associated with the data-transfer and syncing.

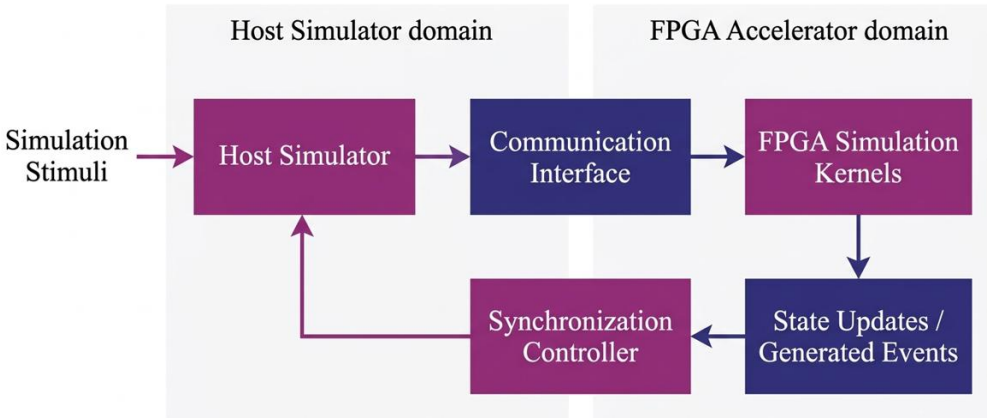


Figure 4- FPGA–host co-simulation data flow illustrating stimulus transfer, accelerated kernel execution, state exchange, event synchronization, and coordinated simulation progression

3.7 Runtime Monitoring and Resource-Aware Reconfiguration

The last architectural element offers a runtime monitoring of the simulating progress and FPGA resource usage. The frame logs records the execution time, accelerator use, rate of communication overhead, regularity of synchronization and distribution of workload. Like measurements can be made to identify inefficient mappings and reconfiguration to an even greater extent.

Resource-conscious management is particularly critical in the case of cloud datacenters as a number of hardware components are likely to share FPGA resources. The framework is thus able to rank kernels based on their computational intensity, anticipated benefit of acceleration and the available logic, memory, and processing resources.

The complete architecture is a cycle of tight execution where the characterization of the workload of the RTL model is connected to profiling, partitioning, FPGA mapping, co-simulation, monitoring and reconfiguring. The design will enable the simulation environment to adapt to different work load characteristics and not necessarily a fixed accelerator set up.

3.8 End-to-End Framework Workflow

The proposed working flow will include loading the RTL design and cloud workload specification. The profiling step will, in turn, establish simulation hotspots and its parallelism and resource requirements. Based on these features, the partitioning mechanism allocates suitable kernel to the FPGA and leaves other tasks to the host. FPGA definition is developed according to the kernels selected and hardware and software simulation interactions are started.

In simulation, the host drives stimulus and control data and the FPGA does some computationally intensive kernels of prediction of interest. The sim-layer organizes exchange of states and ensures consistency in the simulation. The generated results are identified and fed to the analysis layer where they can be evaluated on execution time, acceleration ratio, the FPGA resource utilization, synchronization overheads and scalability. To either suspect the inefficient use of the resources or the overcommunication, the workload mapping can be reconfigured and used at the subsequent stages of the simulation.

In such a way, the proposed architecture has offered a single platform of synthesizing reconfigurable FPGA acceleration, partitioning of RTL and software simulation, as well as hardware-software co-simulation. Its value addition is not simply a faster implementation platform based on an FPGA, but the integration of profiling, adaptive partitioning, configurable hardware execution, synchronization management and cloud workload awareness has been put into a single simulation platform.

IV. EMPIRICAL VALIDATION OF RECONFIGURABLE FPGA–HOST SIMULATION EFFICIENCY

The suggested framework is empirically tested to determine its capability of minimizing RTL simulation latency with no impact on the effective coordination of FPGA-accelerated simulation kernels with simulation functions that can be implemented on the host architecture. Instead of comparing two clouds by way of an experiment, the analysis is made in the context of experimentally verifiable behavior in relation to real workload setups of datacenters. The empirical analysis presupposes the time of simulation running, acceleration ratio, FPGA resources utilization, synchronization overhead, and scaling amongst the major indicators of the performance of the framework. These points allow viewing the efficiency of the suggested architecture of reconfigurable hardware integration with software simulation in terms of multiple dimensions.

4.1 Empirical Assessment Dimensions

The assessment takes into account five dimensions that are directly related to architectural goals of the framework. Execution time of Simulation: The time to execute a simulation workload. The acceleration ratio calculates the ratio of the decrease of the simulation time supported by FPGA execution in reference to the execution under the help of the software. Utilizing FPGA resources will determine the extent to which the accelerated simulated kernels utilise configurable logic, registers, memory resources and other hardware constructs. Synchronization overhead captures the cost of movement of simulation states and events among the FPGA and the host. The observation of the behavior of the framework to larger workload, RTL complexity and heterogeneous hardware components is called scalability.

These are dimensions that are combined together since acceleration on its own does not give a complete portrayal of framework efficiency. An FPGA layout with high computational throughput, but high synchronization or consumption of hardware need not be of much practical use. Consistency of computation, communication, resource usage and workload increase is thus the correlation under analysis in the empirical analysis.

4.2 Simulation Latency and Acceleration Behavior

The delay of the simulation is also experimented by searching the performance of the software-only simulation patterns and FPGA supported simulation patterns. The identification of computationally-intensive RTL through profiling, and the mapping of computationally-intensive to FPGA resources features the proposed architecture, the control-intensive and irregular functions are modeled in the host simulation environment. This partition helps in the sequential algorithms (host processor) by taking out the computational load.

The empirical performance is expected to be particularly good when RTL kernels with repetitive operations, a high degree of event frequency and high parallelism are taken into consideration. This type of kernel can take advantage of simultaneous FPGA execution, and simulation areas dominated by control remain with the freedom of software execution. This hence shows that the acceleration ratio is workload dependent and that is, marks the ratio of the simulation activity that could be successfully targeted to reconfigurable hardware.

This is also the characteristics of a good principle of design: the secrets to successful FPGA acceleration are not in the straight transfer of RTL simulation functions to hardware but in a well-designed division of labor.

4.3 Resource-Aware Framework Behavior

The efficiency of the utilisation of FPGA resources is examined, to find out how well the framework could be used to map the simulation kernels to the configurable hardware. It considers logic resources, registers, embedded memory and accelerator specific buffering requirements in the analysis. The reason is that the framework is not geared towards reaching maximum occupancy of hardware, a reconciliatory balance between the computing capacity and the resources is sought.

There may be varying hardware demands of various cloud workloads. RTL areas that are control intensive may require additional logic and datapath units and simulation kernels that are memory based may require additional buffering and bandwidth. These workload specifics can be reconfigured by reconfiguring the FPGA mapping, which is possible thanks to the reconfigurable architecture.

Practically this provides it with increased flexibility compared to a fixed-function acceleration architecture as the available resources can be reconfigured to suit the simulation bottlenecks which are most important. Acceleration and use of resources is thus perceived as well as a disengaged indicator.

4.4 Synchronization and Host–FPGA Coordination

The FPGA-accelerated kernels and the host simulator interact with each other to empirically assess the behavior of synchronization. These primary ones are state exchange, transfer of event, coordination during simulation stage, and data motion. An over-synchronization can reduce the usefulness of hardware acceleration even to small accelerated kernels, or a very entertainment-oriented kernel.

The proposed framework combats this problem by working on the partitioning of workloads that tries to keep sufficiently the computational region of the FPGA and minimizes superfluous transfers. Synchronization is done at significant execution points instead of on the completion of each individual simulating operation. Using this approach, computationally intensive parts can execute more independently, though are consistent with the host simulation state.

The empirical evaluation that will be the result is the indication of whether the benefit in performance of calculation which is obtained by using FPGA execution will still be higher than the cost of communication and coordination.

4.5 Scalability Across Heterogeneous Workloads

Scalability is studied in terms of simulation settings of increasing scale, and those that are increasingly nonhomogeneous. Scalability dimensions are regarded as size of workloads, RTL complexity, and the number of modeled processing and memory as well as provided accelerator, and interconnect elements.

In jobs whose computing intensity is increasing the pace of the simulation activity that can be handled by an FPGA may increase, and further opportunities of parallel acceleration emerge. However, there are additional hardware requirements and communication directions that can also increase synchronization requirements. The framework is next tested on its ability to sustain a sustainable balance between acceleration and coordination with the greater complexity of workload.

The reconfiguration mechanism also enables scalability with the possibility of remapping FPGA resources in case of a change in the prevailing simulation characteristics. This is taken care of to make it possible to apply the structure to the study of different cloud datacenter architecture in an iterative manner with an un-fixedly deployed accelerator environment.

4.6 Empirical Validation of Simulation Consistency

The consistency of behavior is taken into consideration as well as performance observations. The behavior of FPGA-assisted execution is compared to the software simulation behavior at pertinent simulation boundaries. Consistency is empirically evidence in the sequences of the events, transitions between the states, outcomes of the transctions and the ultimate states of simulation.

This test will ensure that the acceleration will not go beyond the functional behavior that RTL model being tested is. The goal is not simply therefore to found reduced latency in simulation and demonstrate that reconfigurable acceleration can be accommodated alongside with software-based simulation without impacting simulation fidelity.

Overall, the empirical analysis demonstrates the efficiency of the framework conducted in the concerted consideration of simulation latency, acceleration behavior, resource efficiency, synchronization cost, scalability and consistency of simulations. Such a multidimensional viewpoint mirrors practical needs of FPGA-assisted RTL simulation and offers a platform to analyze the framework as a generalized architecture to workload modeling and hardware-software architecture exploration of cloud datacenters.

V. FUTURE ENHANCEMENTS AND ADAPTIVE EVOLUTION OF THE FPGA CO-SIMULATION FRAMEWORK

The provided framework provides the foundations of accelerated RTL simulation on the premise of reconfigurable FPGA-host cooperation but several extensions can guarantee the improved level of flexibility, scaling and execution to more complex cloud datacenter architecture models.

5.1 AI-Guided Workload Partitioning

Machine learning can be used in future versions and automatically detect areas of simulation appropriateness and decide whether particular RTL kernels are to be executed by the FPGA or the host processor. The historical simulation properties, frequency of execution, intensity of communication and resource needs can be developed to workload-partitioning modeling. All decisions guided by AI can lead to the active placement of the hardware/software interface in accordance with the dynamics of the workload and the amount of manual configuration operations can be reduced.

5.2 Runtime Partial Reconfiguration

Partial reconfiguration of run time of FPGA is a significant avenue towards efficiency in terms of resource usage. The accelerator could dynamically scale up particular areas rather than fully issuing particular configurations to the entire FPGA to support the individual workloads. This would enable the framework to fulfill the needs of numerous heterogeneous datacenter workloads on to a common FPGA base and minimize the effect of configuration.

5.3 Distributed FPGA Co-Simulation

The model can be extended to distributed FPGA-based simulation wherein two or more FPGA centres work together over cloud or datacenter environments. Large RTL models could be fortuitously partitioned into several acceleration nodes by the dependencies of both communication and computing intensity. Communication would have to be limited and effective inter-FPGA timing systems would be required to guarantee similarity in both simulation and limiting.

5.4 Advanced Memory and Interconnect Optimization

High-bandwidth memory, smart buffering, and smart FPGA-host interconnects can be explored as part of future development to minimise data-transfer latency. New interfaces, and new memory topologies, can bring more often-used simulation state to be nearer to accelerated kernels, leading to better adoption of accelerator resources.

5.5 Automated Design-Space Exploration

A design-space exploration layer can be automatically scanned to experiment with other partitions of the RTL code, mappings to FPGA parts, synchronization granularities, and workload distributions. Such a mechanism would be in a position to calculate resource efficient design of different cloud datacenter structures without having to do a lot of manual optimization.

5.6 Broader Datacenter Workload Modeling

The framework can be extended to AI training and inference, distributed analytics, workloads requiring high storage and high-speed network scenarios in the future. With the integration of these classes of workloads, the framework would be better able to support heterogeneous cloud platforms, and enable exploration of hardware-software architecture in a more comprehensive manner.

VI. CONCLUSION

In the paper, a reconfigurable FPGA-based platform was outlined which accelerates RTL simulation and hardware-software co-simulation of heterogeneous workloads of cloud datacenters. It can be applied to address the increasing computational demands of highly detailed simulation of processors, accelerators, memory subsystems, and high-speed interconnects by utilizing the scalability of custom software simulation, and the real time capabilities of FPGA resources. The framework does not transfer the whole simulation setup to hardware, but rather chooses suitable kernels to customable FPGA resources, depending on factors of the simulation focus of computational intensity.

This design involves the RTL workload characterization, simulation profiling, adaptive workload partition, FPGA-based kernel acceleration, host-side simulation control, resource aware reconfiguration, and synchronization management. This organization defines a cooperative execution paradigm with the FPGA running computationally intensive and parallelizable simulation workloads and the host computer running control-based irregular coordination sensitive workloads. This partitioning has the flexibility to support various RTL designs and heterogeneous cloud workload properties.

The methodology of the empirical evaluation takes into account the simulation execution time, acceleration ratio, use of resources in the FPGA, synchronization overhead, scalability and consistency of simulation. These dimensions give a holistic foundation towards the measurement of behavior in the frameworks since successful acceleration is not just based on computational throughput but also resource efficiency and coordination between the host and FPGAs. The framework specifically aims at workloads that have repetitive and strictly active, parallelizable simulation kernels, and configurable hardware is able to allow higher execution concurrency.

The solution offered is also useful in establishing a stepping stone towards future enhancement of AI-based workload allocation, runtime partial reconfigurations, distributed FPGA Co-simulation, and highly developed memory and interconnect architectures and automated design space exploration. These would complement the existing customizability of the simulator with FPGA support to an ever-increasing number of different datacenter platforms.

The framework, on the whole, demonstrates how one line of architecture can be constructed to cut down the overhead in any software development and how to also cut down differentiation, at the same time (and consistency in simulation). It provides a reconfigurable hardware accelerator, which is integrated alongside adaptive workload (partitioning) and host-execution (coordination) to provide a more flexible base of faster hardware-software design validation and architecture exploration in cloud datacenter systems.

REFERENCES

[1] Karandikar, S., Mao, H., Kim, D., Biancolin, D., Amid, A., Lee, D., Pemberton, N., Amaro, E., Schmidt, C., Chopra, A., Huang, Q., Kovacs, K., Nikolic, B., Katz, R. H., Bachrach, J., & Asanović, K. (2019). FireSim: FPGA-accelerated cycle-exact scale-out system simulation in the public cloud. *IEEE Micro*, *39*(3), 56–65. https://doi.org/10.1109/MM.2019.2910175

[2] Beamer, S. (2020). A case for accelerating software RTL simulation. *IEEE Micro*, *40*(4), 112–119. https://doi.org/10.1109/MM.2020.2997639

[3] Amid, A., Biancolin, D., Gonzalez, A., Grubb, D., Karandikar, S., Liew, H., Magyar, A., Mao, H., Ou, A., Schmidt, C., Wright, B., Shao, Y. S., Wright, B., Nikolić, B., & Asanović, K. (2020). Chipyard: Integrated design, simulation, and implementation framework for custom SoCs. *IEEE Micro*, *40*(4), 10–21. https://doi.org/10.1109/MM.2020.2996616

[4] Elsabbagh, F., Sheikhha, S., Ying, V. A., Nguyen, Q. M., Emer, J. S., & Sánchez, D. (2023). Accelerating RTL simulation with hardware-software co-design. In *Proceedings of the 56th Annual IEEE/ACM International Symposium on Microarchitecture* (pp. 153–166). Association for Computing Machinery. https://doi.org/10.1145/3613424.3614257

[5] Wang, H., & Beamer, S. (2023). RepCut: Superlinear parallel RTL simulation with replication-aided partitioning. In *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems* (Vol. 2, pp. 219–232). Association for Computing Machinery. https://doi.org/10.1145/3582016.3582034

[6] Zhou, K., Liang, Y., Lin, Y., Wang, R., & Huang, R. (2023). Khronos: Fusing memory access for improved hardware RTL simulation. In *Proceedings of the 56th Annual IEEE/ACM International Symposium on Microarchitecture* (pp. 411–424). Association for Computing Machinery. https://doi.org/10.1145/3613424.3614275

[7] López-Paradís, G., Li, B., Armejach, A., Wallentowitz, S., & Moretó, M. (2023). Fast behavioural RTL simulation of 10B transistor SoC designs with Metro-MPI. In *2023 Design, Automation & Test in Europe Conference & Exhibition (DATE)* (pp. 1–6). IEEE. https://doi.org/10.23919/DATE56975.2023.10137080

[8] Lin, D.-L., Ren, H., Zhang, Y., Khailany, B., & Huang, T.-W. (2022). From RTL to CUDA: A GPU acceleration flow for RTL simulation with batch stimulus. In *Proceedings of the 51st International Conference on Parallel Processing* (Article 91, pp. 1–11). Association for Computing Machinery. https://doi.org/10.1145/3545008.3545091

[9] Chatterjee, D., DeOrio, A., & Bertacco, V. (2011). Gate-level simulation with GPU computing. *ACM Transactions on Design Automation of Electronic Systems*, *16*(3), Article 30. https://doi.org/10.1145/1970353.1970363

[10] Babb, J., Tessier, R., Dahl, M., Hanono, S., Hoki, D. M., & Agarwal, A. (1997). Logic emulation with virtual wires. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, *16*(6), 609–626. https://doi.org/10.1109/43.640619

[11] Ahmad, T. B., Kim, N., Min, B., Kalia, A., Ciesielski, M., & Yang, S. (2012). Scalable parallel event-driven HDL simulation for multi-cores. In *2012 International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)* (pp. 205–208). IEEE. https://doi.org/10.1109/SMACD.2012.6339456

[12] Qian, H., & Deng, Y. (2011). Accelerating RTL simulation with GPUs. In *Proceedings of the 2011 IEEE/ACM International Conference on Computer-Aided Design* (pp. 752–757). IEEE. https://doi.org/10.1145/2051118.2051187

[13] Nanjundappa, M., Patel, H. D., Jose, B. A., & Shukla, S. K. (2010). SCGPSim: A fast SystemC simulator on GPUs. In *Proceedings of the 15th Asia and South Pacific Design Automation Conference (ASP-DAC 2010)* (pp. 145–150). IEEE. https://doi.org/10.1109/ASPDAC.2010.5419903

[14] Miller, J. E., Kasture, H., Kurian, G., Gruenwald, C., Beckmann, N., Celio, C., Eastep, J., & Agarwal, A. (2010). Graphite: A distributed parallel simulator for multicores. In *HPCA - 16 2010 IEEE 16th International Symposium on High Performance Computer Architecture* (pp. 1–12). IEEE. https://doi.org/10.1109/HPCA.2010.5416635

[15] Chen, J., Annavaram, M., & Dubois, M. (2009). SlackSim: A platform for parallel simulations of CMPs on CMPs. *ACM SIGARCH Computer Architecture News*, *37*(2), 20–29. https://doi.org/10.1145/1555754.1555781

[16] Tan, Z., Qian, Z., Chen, X., Asanović, K., & Patterson, D. A. (2015). DIABLO: A warehouse-scale computer network simulator using FPGAs. In *Proceedings of the 20th International Conference on Architectural Support for Programming Languages and Operating Systems* (pp. 331–344). Association for Computing Machinery. https://doi.org/10.1145/2694344.2694362