COMMUNICATION EFFICIENCY AND PATTERN OPTIMIZATION IN FPGA ACCELERATED RTL SIMULATION

Steffen Köhler, Jan Schirok, Rainer G. Spallek

Institute of Computer Engineering
Technische Universität Dresden
D-01062 Dresden, Germany

stk@ite.inf.tu-dresden.de(Steffen Köhler)

Abstract

For verification of complex RTL circuit models, simulation efficiency can be increased through the application of FPGA accelerators and their integration into software-based environments. However, this integration requires both RTL model partitioning and data exchange between these partitions mapped either to fast FPGA hardware or to traditional software based RTL simulators. Beside the hardware execution benefits, the acceleration degree of the entire simulation process heavily depends on the organization of the inter-partition communication scheme, which is strongly related to the particular transfer patterns. In this paper, a novel, highly efficient communication scheme named interaction-based modeling is introduced and evaluated against widely used non-overlapping and transaction level modeling (TLM [1]) methods. Applying the new model to typical communication patterns mainly found in stream processing applications, we could decrease the number of transfered events, thus gaining an overall simulation speedup over the simple communication model. In terms of simulation parallelization and acceleration, the interaction-based modeling can compete with TLM methods, without requiring a high modeling effort. In contrast to TLM, the interaction-based communication provides a versatile and less abstract interface model, which allows more flexible partitioning and requires significantly less FPGA resources. Real-world simulation results have been obtained from a RTL simulation prototype platform, which is also described in this paper.

Keywords: RTL Simulation, FPGA Accelerator, Communication, Partitioning

Presenting Author's Biography

Steffen Köhler is a research and teaching staff member of the Department of Computer Science at the Technische Universität Dresden. He received his Diploma (Dipl.-Inform.) in Computer Science and his Doctoral degree (Dr.-Ing.) in Computer Engineering in 1992 and 2009 respectively. His research interests include the analysis and optimization of data and instruction streams in embedded processors as well as trace data reconstruction and simulation.

1 Introduction

The ever growing complexity of VLSI systems raises the demand to accelerate RTL-based simulations. Beside the concept of complete FPGA-based emulation, the migration of partitions of a RTL model to FPGA hardware can be considered a promising alternative. In contrast to expensive emulation platforms, less sophisticated FPGA-based systems and prototyping platforms are a cost effective solution for simulation acceleration in small to medium scale VLSI designs. These systems are easily integrated into commercially available RTL simulators, thus improving simulation speeds within a familiar working environment. As a result, the design cycle will be shortened significantly, allowing a faster achievement of design goals and time-to-market requirements.

The partitioning of the hardware design, however, introduces an additional step to the design process, which may complicate the FPGA-based acceleration of RTL simulations. The suitability and flexibility of the communication interface for a sufficiently large class of partitioning tasks is essential for an efficient mapping of RTL design partitions to FPGA hardware. The achievement of a highly-parallel operation of all partitions can be considered a main goal of our coupling approach. Thus, it is essential to keep the communication latency low, while trying to maximally overlap transfers to hide latencies. To reduce the amount of inter-partition data to be transfered, the communication protocol has to be kept as simple as possible. Complex transaction oriented communication (TLM) significantly limits partitioning possibilities and requires an explicit design of an abstract communication model.

Besides timing information, our proposed interaction-based method does not further encapsulate transfered data, so as to require a less complex signal coding and decoding scheme within the FPGA hardware and simulation software. Since it provides a less abstract interface, the interaction based method shows a reduced partitioning and generalization effort as compared to TLM approaches, while achieving an almost equal simulation efficiency, but without requiring costly coding and decoding blocks.

Possible disadvantages of inefficient partition coupling should be especially avoided when high data-rate, flow-oriented RTL components have to be verified by using off-the-shelf simulation tools. This can be considered a typical case of media and telecommunication processing applications. In our paper, we therefore extract relevant communication patterns that can be adapted to particular partitioning and simulation tasks. Through a formalization of directional initiations and reactions of a simple, non-overlapping communication channel, a new parametrizable communication model is created and evaluated.

2 Related Work

Improving the performance of RTL simulations using tailored hardware is established both in commercially available emulator devices as well as in smaller prototyping platforms. Emulators are for example Mentor Graphics Veloce [2] and Cadence Incisive Palladium [3]. These devices have a broad range of applications and have great speedups compared to completely software-based simulators. Apparently, the mentioned emulators are high-end and also cost-intense.

Less expensive hardware solutions incorporate a FPGA based hardware accelerator which is coupled to a PC running the software-based simulator. A common way to integrate the FPGA hardware is the use of standard system interfaces such as PCI. Simple interface concepts [4] do not pay much attention to the communication scheme. As the output pattern of the design under verification (DUV) highly influenced the simulation acceleration, a proper partitioning was critical for improvements of simulation speed. Data exchange between the two partitions was only done on changes of the signals, effectively limiting the amount of data transferred.

Other FPGA-based solutions exist, which provide a special clock acceleration mode [5]. Through this technique, the data exchange between DUV and host PC can be skipped during certain simulation periods. A significant speeding up of the simulation process can be achieved in most cases, provided the fact that the periods to skip are known to the developer. No formalism is provided to automatically find those periods which effectively limits the use for this method to special cases where a lot of knowledge about the simulated components must be provided for large speedups with accurate simulation results.

When debugging hardware designs, short turn-around-times, e.g. times for one test-run with changed HDL sources including test-time and synthesis, are vital for a productive working environment. This would imply the avoidance of hardware re-compilation in as many cases as possible [6]. Using generic interfaces between the designed entities, most modules can be left in hardware while some modules, for which the HDL source has changed, may be simulated in software. The generic interfaces provide a possibility to change the actual location of each module, whether simulated in hardware or software while not changing the overall working model. The more modules will be run in software, the less speedup for a test-run can be accomplished, but no additional synthesis times are needed in this case. The use of the generic interfaces is essential for this kind of simulation, either by using them in the first place or by their later automated addition.

A further interesting way of debugging hardware designs is the replication of a simulated design entity in an FPGA [7]. The original design is directly verified against a simulation testbench, while for the replicated design the inputs are delayed by N clock cycles. If an error occurs in the design, which can be triggered by the original design, its cause can be traced back in the replicated design. The solution highly depends on the knowledge of the FPGA internals in order to read out the state of the replicated design. Especially for long

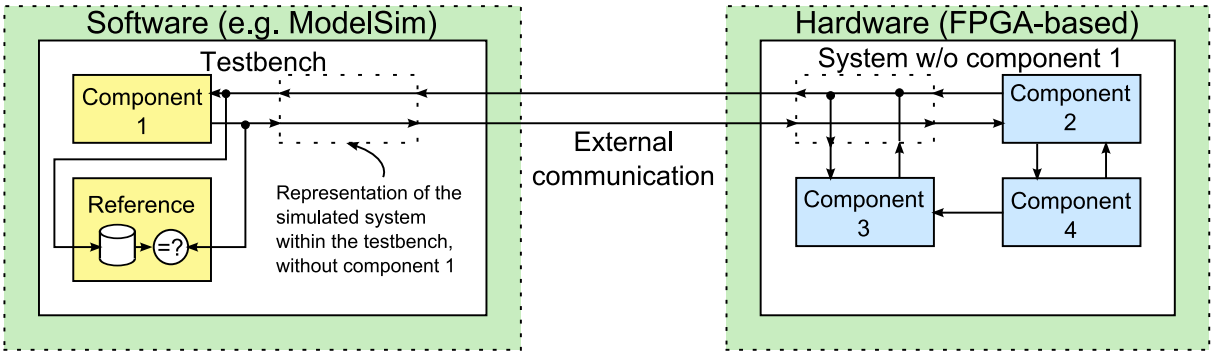

Fig. 1 Interaction-based Communication Scheme

simulation times, the described method can be applied, giving a high simulation speed during the initial phase until the error occurs.

3 Interaction-Based Communication

For a problem outline, assume a partitioning into two design parts each containing circuit components to be verified and a simulation flow on register transfer level (RTL). This exchanges data between the partitions using simple binary logic levels (ones and zeros), which might be a limitation for handling multi-value representations including uncertain or tri-state values. As explained later, this is not a significant limitation of our concept because only binary logic levels will represent meaningful data.

For the interaction based communication, the data exchanged between design partitions remains unchanged. In particular, the transferred bits will not be collected, encapsulated or emitted by applying a special model. Using TLM would be an example for the opposite approach, where a communication model is used on both sides, which requires transforming the low-level interface data into a high-level transaction. An overview of our proposed system is given in figure 1.

An efficient way of communication between the partitions during simulation would be the identification and the removal of any unnecessary transfers. Additional optimization potential can be utilized by bundling adjacent transmissions. For both approaches, a declaration of unnecessity is required. A good approach for the unnecessity can be obtained when considering both partitions as finite state machines (FSMs). A data exchange has to be performed in the current simulation clock cycle only when a state transition inside the reacting partition is sensitive to the input of the initiating partition. On the other hand, if the state transitions inside the reacting partition does not depend on interface data, the related transferred data can be omitted in this cycle, thus already ignoring the available interface data inside the initiating partition.

Since partitioning is mainly driven by already existing component boundaries, the extraction of input sensitivity has to be obtained from a predefined interface structure. This is usually possible in data flow applications where existing data flow control signals can be identified and re-used as transfer indicators.

A common example for neglecting unknown or undefined data value in transmissions is a simple FIFO communication scheme, which implements buffering of the applied data when indicated by a special handshake signal. If this signal is not asserted, the data transmission can be omitted by the initiating partition. As a result, unknown inter-partition transfer values can be ignored if there is enough certainty that the handshake signal is de-asserted so that all data including unknown or undefined values would not be propagated. In all other cases, e.g. RTL interface model errors, simulation accuracy is reduced and a warning message will be issued during simulation. For FIFO interfaces, data propagation flag signals can be automatically detected by netlist decomposition methods (Shannon Expansion), which are however not subject of this paper.

Because data is transferred from one partition to another in a bidirectionally overlapping manner, we will introduce two simple terms *actio* and *reactio*, just to distinguish the two ways of communication. *actio* should stand for output sets of an initiating partition, whose transmission is required because it triggers a state transition inside the reacting partition. While the reacting partition is responsive to the initiating partition, it may also act as an initiating partition for a backward communication with a resulting state transition, which is called *reactio* in this paper.

A further requisite for the communication between the partitions is required in order to achieve high simulation efficiency. In our proposed simulation framework it is assumed that no unexpected *reactio* is sent. An unexpected *reactio* is for example an (unexpected) interrupt from the reacting partition. For every *reactio*, the exact timing relation to all corresponding *actio* has to be known exactly. If this cannot be granted, the interaction-based communication scheme will temporarily fall back to a simple, non-overlapping, inter-partition transfer. This problem, which slightly narrows the possible application range, for which a high simulation efficiency is obtainable, is also present in TLM approaches.

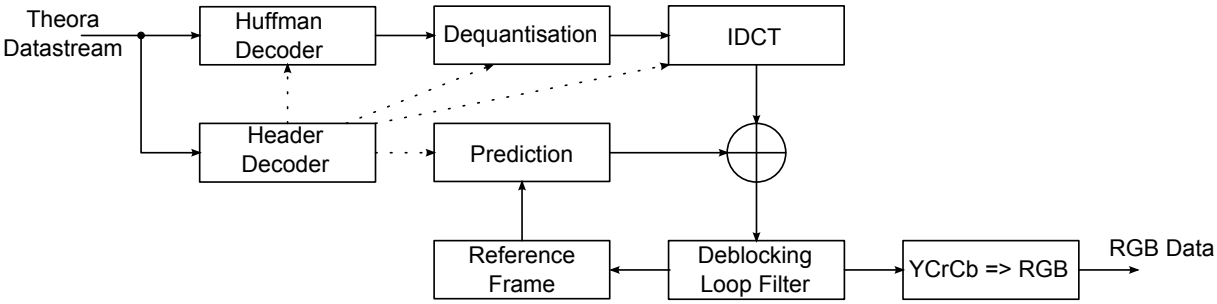

Fig. 2 Theora dataflow

Typical applications, where no unexpected *reactio* has to be sent can be found in the area of media stream processing and telecommunication algorithms. A directional data flow allows simple partitioning along the data flow path. A second example for the absence of an unexpected *reactio* are basic bus devices like RAM or coupled peripherals, which only send a *reactio* (read out data) for a certain *actio* (data access). Directional debug communication, e.g. printf-debugging or the behavior of logic analyzers, is another example, for which the simulation efficiency can be improved because of no unexpected *reactio*.

Through the application of the described techniques, a simulation speed improvement can be obtained. The speedup can be further increased by bundling the *actio* associated data of consecutive clock cycles, provided that there is no *reactio* during these cycles. The maximum number of cycles, for which data is transferred continuously between partitions of the simulation framework, is called *nread* in our proposed flow. A second possibility for speed improvements is the prevention of any data transmissions during phases, for which no *actio* is emitted. When the next *actio* is about to be sent, a special indication message is inserted into the transmitted stream. This message encodes the cycle, at which the *actio* has been issued. The maximum number of cycles, which are skipped by the simulation process, is called *nskip* in our framework.

4 Demonstration Platform

A prototype for demonstrating our simulation flow has been created using a PCI-express FPGA add-on card [8], which comprises the hardware partition, and Mentor Graphics ModelSim for the software partition. Low level interfacing on the software side was achieved through a custom kernel device driver providing board access through the underlying Linux OS and the ModelSim FLI interface [9]. The already mentioned parameters *nskip* and *nread* can be determined and adjusted at simulation runtime within the software-based partition, thus providing direct and interactive control of the communication interface. However, in our shown example the parameters were kept constant during a simulation run.

The model chosen for simulation was a hardware accelerated video decoding application described in [10], including both an embedded LEON3 processor core and a connected dedicated video decoder component. Decoding is performed according to the Open Theora standard [11]. The described system uses a publically available hardware/software build environment [12], that includes all required components. The model was chosen because of its relevance to audio/video media processing and its open source nature. A simplified dataflow for the video decoder can be found in figure 2.

A major block of the video decoder accelerator is the inverse discrete cosine transform (IDCT), which was chosen to demonstrate the hardware-accelerated component verification process at RTL level. To achieve a partitioned simulation, we isolated this block at RTL level to fully analyze it using ModelSim, while leaving the remaining SoC component's RTL description inside the FPGA partition. The idea behind this methodology is to evaluate whether the IDCT works as expected within the entire system context. By keeping the system components unchanged in the FPGA partition, changes to the IDCT can be made more easily, as the modifications to the IDCT do not require a re-run of the FPGA design-flow. A re-compilation within the ModelSim RTL partition is sufficient to restart the simulation process. An overview for the prototype system partitioning is shown in figure 3.

For the communication between the IDCT component and the remaining system, two unidirectional synchronous stream interfaces (FIFOs) can be applied. Inter-partition communication is performed fully synchronous with fixed input, processing and output the clocking schemes. For this type of a commonly used stream interface, the calculation of *actio* and *reactio* timings can be directly derived from the static predefined interface timings. The hardware partition initiates a computation within the IDCT component by sending data to it (*actio* initiated by a write enable signal). After a fixed amount of clock cycles has elapsed, the result (*reactio*) is read back. As this read back action is driven by an enable signal as well, it is itself also considered an *actio*. If the number of computation clock cycles is unknown at interface design time or varying (variable pipeline length), an estimate has to be used to select a proper computation clock cycle range which does not diminish the overall speedup of the simulation. A worst case parameterization scenario would be an awaited *reactio* in every cycle until all data has been transferred.

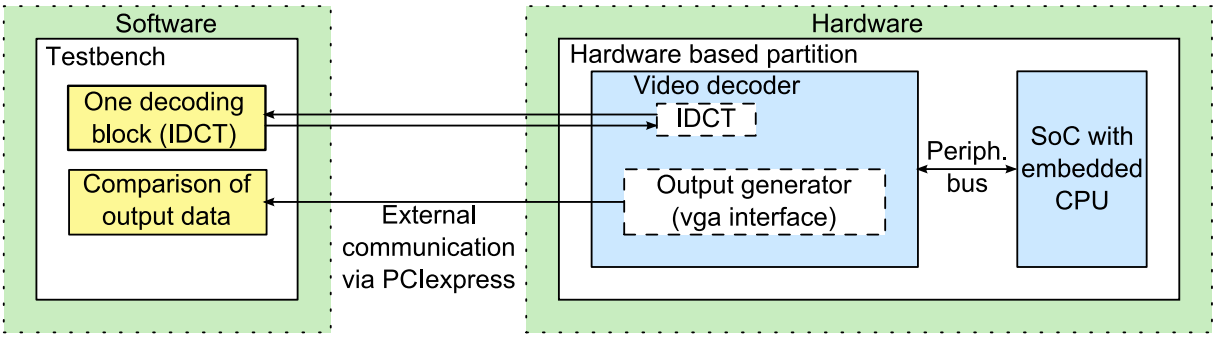

Fig. 3 Prototype System Overview

The speedup for this corner case using non-overlapping cycle-based communication is examined within section 5.

After the simulation starts, the hardware platform cycles the hardware partition, until either *nskip* or a *actio* is reached, which itself is transmitted to ModelSim. A maximum of *nread* data cycles are transferred at once in order to reduce the overhead per cycle. The overall simulation control remains within the software-based simulator, with the hardware platform clocking ahead as many cycles as possible while maintaining correctness.

An amount of less than *nread* data words is transferred when a *reactio* is awaited. This happens if calculation results are expected from the IDCT. Subsequently, the software partition transfers cycle-accurate data to the hardware partition, e.g. the result data of the IDCT, which is now available for further processing by the SoC within the FPGA. The complete calculation of the IDCT outputs is done within ModelSim and can easily be debugged and changed without the need of a resynthesis as long as changes are only made within the software-based partition. If – within ModelSim – an unexpected *reactio* is detected, which itself is a violation of the interface specification of the simulated module, an error message will be issued stating that the correctness of the simulation can no longer be maintained.

The described scenario is one representative example partitioning for a single SoC. Other partitionings, with more complex interface schemes are possible, but unknown or unpredictable inter-partition communication patterns complicate the prediction of achievable performance of the software-based simulation partition. Another limitation, which is known but not examined within this paper, is the propagation of unknown (e.g. X-)values over interface boundaries. Despite the fact, that it should be generally avoided to apply any unknown values to interface signals of modules or components, the correctness of a partitioned simulation is not maintained when transferring unknown values on inter-partition I/O signal lines within our prototype environment. The determination, whether an *actio* is sent or a *reactio* is expected, cannot be made without a fully known interface state.

5 FPGA Prototype Results

The proposed interaction-based communication method was compared to non-overlapping cycle-based communication and to a TLM approach in terms of simulation speed, implementation effort and applicability. The chosen scenario was the former described verification of a signal processing block of the video decoder within the software partition, while the rest of the embedded platform was simulated in the hardware partition. This partitioning was chosen, because the evaluation of a certain building block within the complex processing flow can be considered a common verification case. Additionally, many other approaches show enough flexibility to be adapted to this scenario, with varying implementation effort needed.

Tab. 1 Simulation results

Approach	Simulation $cycles \cdot s^{-1}$
Cycle-based	42,000
Interaction-based (minimal)	43,000
Interaction-based (optimal)	178,000
TLM	200,000

The results for all approaches are summarized in table 1. As previously mentioned, an increase in simulation speed was achieved, with the non-overlapping cycle-based communication being the slowest approach. The speedup for the interaction-based communication is between 1 and 4.2 depending on the used parameters described above. Using optimal values for the parameters the interaction-based communication has a speedup which is only slightly smaller than for the TLM approach (4.8). As already said, TLM is considered less flexible and requires additional implementation effort for the bus functional model (BFM). The cycle-based approach in contrast, is the most flexible by means of communication. It does not demand any additional implementation effort but shows the lowest simulation speeds. The interaction-based method positions itself in between the former named approaches for both speed and implementation effort needed. Through its parameterization, it can bridge the gap between the existing approaches and can be flexibly adapted to a larger number of RTL model interface classes.

The implementation effort for setting up a partitioned simulation is not quantified within table 1, because these values highly depend on already implemented testcases, experience of the simulation engineer and the degree of automation within the simulation environment. In the explained example, the implementation and parameterization of the interface took about 4 to 5 times less design effort when using the interaction-based communication model compared to a TLM approach, while achieving the same level of simulation performance. This effort does not include the partitioning itself and the design of the underlying communication components, which can be considered an inherently re-usable template for other projects utilizing the proposed simulation platform.

6 Conclusions

Interaction-based communication has been shown a promising way for partitioning hardware accelerated RTL simulations. Its interface design methodology provides a customizable communication scheme, that can be adapted to the real communication pattern through parameterization at simulation runtime. Despite of the capabilities of interaction-based communication, several issues remain unsolved, that should be addressed in the future.

First of all, hardware modularization has to be increased to avoid the invocation of the entire design cycle when changes are made to the hardware partition.

Furthermore, the identification of I/O handshake signals from the RTL netlist (Shannon Expansion) would simplify the partitioned RTL simulation flow significantly. Detailed information about I/O transfer control signals would provide a way to automatically evaluate component/module interface candidates for simulation partitioning.

In this paper, we have limited our investigations to stream processing algorithms, which usually show a nearly constant processing time. However, an adaption of *nskip* and *nread* under rapidly changing communication patterns is easily achievable, providing different levels of interaction during the simulation process.

7 References

[1] F. Ghenassia. *Transaction-Level Modeling with SystemC: TLM Concepts and Applications for Embedded Systems*. Springer, New York, 2006.

[2] Mentor Graphics Emulation Systems, 2010. http://www.mentor.com/products/fv/emulation-systems/.

[3] Cadence Incisive Palladium Series, 2010. http://www.cadence.com/products/sd/palladium_series.

[4] M.N. Wageeh, A.M. Wahba, A.M. Salem, and M.A. Sheirah. FPGA based accelerator for functional simulation. *Circuits and Systems, 2004. ISCAS '04. Proceedings of the 2004 International Symposium on*, 5:V–317–V–320 Vol.5, May 2004.

[5] Stefan Reichör. Turbo für die Simulation. *Elektronik SoC*, 2006.

[6] Kyuho Shim, Kesava Talupuru, Maciej Ciesielski, and Seiyang Yang. Simulation Acceleration with HW Re-Compilation Avoidance. In *VLSID '08: Proceedings of the 21st International Conference on VLSI Design*, pages 487–491, Washington, DC, USA, 2008. IEEE Computer Society.

[7] Mario Larouche. Infusing Speed and Visibility into ASIC Verification. White paper, Synplicity, Inc., 2007.

[8] Altera Corporation. *Stratix II GX PCI Express Development Board Reference Manual*, 2007. http://www.altera.com/literature/manual/mnl-s2gx-pci-express-devkit.pdf.

[9] Mentor Graphics. *ModelSim SE Foreign Language Interface Manual*, Software Version 6.3c edition, September 2007.

[10] Xiph.org Foundation. *Theora FPGA Sourcecode*, 2008. http://svn.xiph.org/trunk/theora-fpga/.

[11] Xiph.org Foundation. *Theora Specification*, April 2008.

[12] Andrá Luiz Nazareth da Costa and Timothy B. Terriberry. Hardware Implementation of Theora Decoding, Integration with LEON3, 2007. http://www.students.ic.unicamp.br/˜ra031198/theora_hardware/.