On RTL to TLM Abstraction to Benefit Simulation Performance and Modeling Productivity in NoC Design Exploration

Sven Alexander Horsinka, Rolf Meyer, Jan Wagner, Rainer Buchty and Mladen Berekovic
TU Braunschweig, D-38106 Braunschweig, Germany
Email: {shorsinka, meyer, wagner, buchty, berekovic}@c3e.cs.tu-bs.de

***Abstract*—Growing demand to integrate more functionality into single-chip solutions require new network-based interconnection models (NoC). The resulting increase in design complexity and strict time-to-market restrictions endanger future viability of Register Transfer Level (RTL) centric design processes. To counteract these developments, the abstract design methodologies presented by Transaction Level Modeling (TLM 2.0/SystemC) are gaining popularity. With this paper, we demonstrate the benefits of raising the abstraction level by creating an adjustable NoC simulation model, satisfying the diverse needs of software and system engineers. Based on a proven and tested RTL NoC design, we applied modeling methods defined in the TLM 2.0 standard, creating flexible simulation model. It provides high timing accuracy, enabling precise behavioral and performance analysis. In addition, higher simulation speeds are achieved by adjusting the timing accuracy. The results demonstrate the advantages of variable simulation accuracy: simulation runs are accelerated by more than two orders of magnitude with performance and behavior assessment exposing a limited latency error of less than four clock cycles compared to the RTL model.**

I. Introduction

By further shrinking transistor feature sizes, the available integration density increases steadily. But since 2007, the ITRS observes that the utilized transistor density scaling slowed down to factor of 1.6 per iteration instead of the traditional increase by a factor of 2.0 [**?**]. This productivity gap illustrates the incapability of established hardware/software design processes in exploiting the potential of new fabrication techniques. The call for raised modeling abstraction is increasing as methodologies based on Register-Transfer Level (RTL) struggle to address the ever growing design space. Through this, Transaction Level Modeling (TLM) emerged and continues to gain popularity.

To allow higher design productivity it is necessary to raise the simulation performance. This enables creation of early prototypes, allowing software development and efficient design-space exploration (DSE). As the simulation effort directly correlates with its accuracy, it is sensible to adjust the accuracy according to the specific simulation needs. Greater performance is of significant benefit for software development, whereas system architects require high accuracy to asses behavior and performance of a hardware model.

The research leading to these results has received funding from the ARTEMIS Joint Undertaking under grant agreement n 621429 and from the German Federal Ministry of Education and Research (BMBF), funding contract 01IS14002O.

In this paper, we present an abstraction approach concerned with Network-on-Chip (NoC) interconnection structures. Techniques for modeling bus systems with adjustable timing accuracy are described in the SystemC TLM-2.0 IEEE 1666-2011 Standard [**?**]. We demonstrate how these techniques can be translated to NoC structures for achieving high performance and accuracy simulations.

By abstracting a separately developed and tested RTL NoC design [**?**], we enable fair comparisons of the simulation behavior for all abstraction levels. Our SystemC/TLM 2.0 NoC model allows high-performance simulations over a magnitude faster than the RTL implementation as well as enabling high-accuracy simulations with a limited end-to-end latency error less than three clock cycles when required.

This paper is organized as follows: Section 2 presents related approaches of NoC simulation acceleration. A brief overview of the reference NoC design, timing accuracy in TLM 2.0, and general RTL-to-TLM abstraction goals is given in Section 3. In Section 4, we document the application of different modeling techniques to our reference design. Section 5 compares the simulation behavior of the models regarding performance and accuracy. The paper is concluded in Section 6.

II. Related Work

The MPSoC design space is significantly extended by the application of versatile on-chip network structures. To this end, great effort is put into the development of dedicated NoC simulators. Of these, several are implemented using SystemC and TLM modeling techniques, e.g., PPNOCS [**?**], NIRGAM [**?**], NOXIM [**?**] and NOSTRUM as part of NNSE [**?**]. These tools allow exploration of different network topologies, routing algorithms, buffer sizes, and virtual-channel techniques. Based on user-defined characteristics, the simulation is executed, providing various functional and nonfunctional information regarding performance, latency, and power consumption, making these tools useful for analyzing the impact of different traffic patterns on a chosen design. It is however complicated to assess the actual timing accuracy and simulation performance, as these general-purpose simulators are typically not based on synthesisable hardware models.

The solution presented in NAXIM [?] combines a SystemC-based NoC simulator with the open-source QEMU CPU emulator. By executing the CPU emulators and the network simulation in separate processes, NAXIM takes advantage of multi-core and cluster environments. A high simulation performance is achieved at the cost of timing accuracy.

A cycle-accurate NoC simulator is presented in BookSim [?], but a direct evaluation of the simulation performance against an RTL implementation is lacking.

In contrast, our work aims at creating a complete SystemC/TLM 2.0 platform providing a variable timing accuracy enabling fast and accurate simulations. Manually abstracted from the RTL network design defined in [?], we developed an adjustable TLM router design, significantly increasing the simulation performance.

Methods for automatically abstracting existing RTL IP cores are presented in [?]. But due to limitations regarding the language support and abstraction degree results are still not comparable to manual abstraction. A different approach is presented in [?], where the NoC simulation is offloaded to a parameterized FPGA design. While achieving a very high simulation speed, the configuration flexibility is lacking.

Our NoC design will be integrated into the SoCRocket framework [?], allowing generation of versatile virtual platforms accelerating the design-space exploration. All utilized IP-cores are available as approximately- and loosely-timed TL models as well as synthesisable RTL implementations, thereby allowing the usage throughout the complete MPSoC design process.

III. Basics and Background

A. Reference NoC design

In order to establish a fair comparison between the properties of different abstraction layers, a reference NoC model is selected. This RTL model was designed, optimized, and tested separately by a different development team [?]. An exemplary configuration is shown in Figure ??. The system is organized as a two-dimensional mesh consisting of nine interconnected nodes wherein every node is formed by a router in the middle linked to up to four tiles. Tiles themselves can consist of complete subsystems, memory, or any other custom hardware.

The Network is composed of a set of interconnected routers and comprises the following characteristics [?]:

- **Mesh based topology**: Each router provides eight full-duplex links to connect neighboring routers, and up to four uplinks.
- **Wormhole switching**: Packets are subdivided into flow-control units (Flits) of 140 bit each.
- **Virtual channels**: Congestion is reduced by implementing 8 virtual channels per router-input port.
- **Pipelined arbitration**: Flits are further subdivided and transmitted in a fixed sequence to allow waitstate-free forwarding.
- **Source routing**: The path of a data stream is predetermined and allows for resource efficient implementation.
- **Mixed criticality**: A QoS scheme is implemented to guarantee bandwidth requirements for critical data streams.

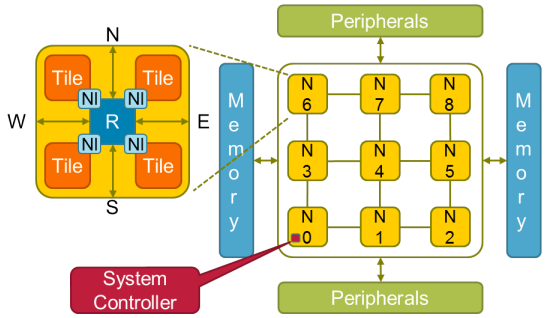

Fig. 1: Configuration of the reference network[?]

The router model is highly optimized with regards to occupied chip area and propagation delays. These steps are necessary when finalizing a RTL model, but result in a higher design complexity and interfere with further development and general maintenance. In addition with the poor simulation performance of large RTL designs, extensive design-space exploration is rendered unfeasible and hence the flexibility of the entire design process is strongly reduced.

B. Transaction Level Modeling (TLM 2.0)

Modeling on the transaction level reserves a certain ambiguity regarding timing annotation. Developers have the freedom of directly catering to specific simulation requirements. Early software development, for example, values high simulation performance over exact timing behavior. In contrast, correct behavior and timing are the main concerns of system developers. To fulfill both needs, the TLM 2.0 standard defines two distinct coding styles [?]:

- **Approximately timed**: High temporal resolution of bus accesses by defining four separate timing points to indicate begin and end of request and response respectively. It allows precise exploration of behavior and performance at the cost of additional simulation overhead.
- **Loosely timed**: Here, a transaction is only defined by its start and end point, thereby reducing the simulation overhead at the cost of precision. This is advantageous for extended simulations and software development.

Both coding styles focus on the temporal resolution of bus accesses. As the allocation of the global communication medium is of high importance for a processing system, directly translating these coding styles to a distributed interconnection network will not yield the desired result. As packet based networks usually have well-defined behavior regarding the length of a single transaction, defining frames, packets or flits with a fixed size reduces the benefits of high-detailed timing points for every transactions.

Looking at the end-to-end behavior of a network, the focus shifts towards the accumulated delay of a package when it reaches its destination. This latency is composed of a fixed component (best-case propagation latency, depending of

the hop count) and a dynamic component (delay caused by congestion). Varying the accuracy of the dynamic component can now be used to achieve a comparable flexibility regarding timing and simulation performance. To capture precise latency informations, the status of the network has to be tested for possible congestion after every transaction (Figure **??**, left). This will obviously result in significant simulation effort, however, when the fixed component of the latency is sufficient, the simulation performance can be greatly increased by ignoring congestion altogether. This allows traversing the whole network in an undisrupted transaction sequence (Figure **??**, right). To avoid confusion, these modeling styles will subsequently be called timed (capturing congestion) and untimed (ignoring congestion) modeling styles.

Methods for estimating the latency caused by congestion without fully modeling it are presented in [**?**] but not currently incorporated in this work.

C. Abstraction goals

The most commonly applied simulation strategy for hardware designs is the event-based simulation model. Tools are available and in use for all major hardware description languages (VHDL, Verilog and SystemC [**?**]). Here, the behavior of a hardware model is represented through a set of discrete events in time. This allows efficient simulations, since in contrast to cycle-based techniques only real state transitions are elaborated. As the simulation performance is a major concern of this work, it is important to identify the properties that contribute most to execution effort. Behavior described on the RTL consists of numerous parallel processes triggered by signal transitions. In so-called delta cycles, the involved processes are evaluated separately and their output signals are synchronized causing other processes to trigger until a stable state is reached. This results in significant runtime overhead consisting of context switches and synchronization [**?**]. By raising the abstraction, the functionality can be expressed through sequential code and thereby reduce the runtime overhead. Among others, this aspect is a important concern when abstracting an RTL model, cumulating in the following TLM design goals [**?**]:

- **Speed**: Raising the simulation performance is necessary for enabling a cost-effective and flexible design process.
- **Accuracy**: Disregarding timing accuracy will result in unreliable simulation results. Depending on the use-case, a compromise between speed and accuracy must be established.
- **Productivity**: Disregarding specific RTL implementation details in early design-space exploration allows for higher productivity and maintainability.

IV. Model Abstraction

A. Data and Timing Abstraction

When abstracting the data representation of an RTL model it is important to identify the right granularity. The smallest individually routeable transfer units will allow a accurate reproduction of the routing and congestion behavior, while minimizing the number of individual transactions. Data on the lowest layer of the presented reference design is organized as 35 bit wide phits. This enables a 4 step pipeline based arbitration parallel to the arrival of a flit. As this computation on the TL can be combined into a single sequential process removing the necessity to subdivide flits. Concentrating the transmission of a flit into a single function call significantly reduces the simulation effort. This also circumvents the need to physical copy the data from one signal vector into another, as only a object reference is passed. The corresponding delay is then annotated as an additional parameter allowing the receiver to synchronize the transaction if needed.

The transfer protocol implemented by the reference RTL model defines precise bit ranges to represent control information regarding the route, virtual channel and other data flow signals. Abstracted to the TL, payload and control information is divided and handled individually, increasing the flexibility and aiding further development. To this end the TLM 2.0 library supplies a basic transport objects (`generic_payload_object` [**?**]) implementing typical bus protocol functionality and the freedom to add custom extensions. Allowing the developer to encapsulate functionality (for example the routing) into specific classes to hide the implementation and thereby allowing modifications without

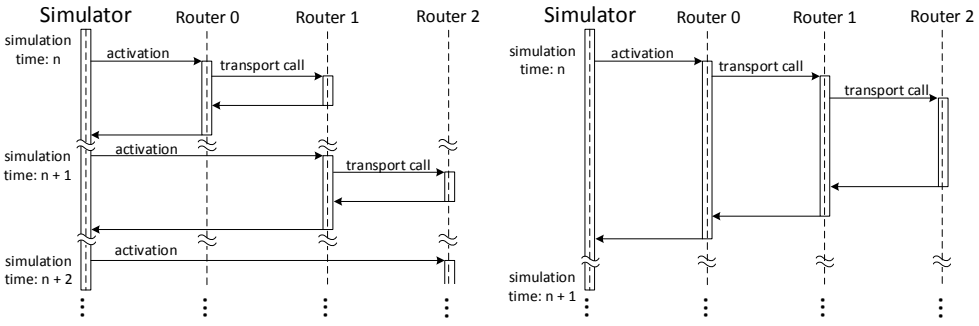

Fig. 2: Synchronization after every hop (timed modeling; left), synchronization after arrival (un timed modeling; right)

influencing other parts of the model. For the efficient handling and reuse of these transport objects it is recommended implement an ad hoc or dedicated memory manager to prevent the costly new allocation for every transaction [**?**].

B. Timed Model

To assess the actual performance and throughput of an on-chip network the simulation model requires a sufficient timing accuracy. The congestion behavior must be reproduced correctly to achieve realistic packet latencies and arrival order. Therefore the simulation model performs a synchronization between competing data streams after every hop to conduct the arbitration and QoS selection (see Section **??**). To reduce the runtime overhead, our approach implements a queue based synchronization only requiring a single thread per router. The explicit coordination between threads is avoided by invalidating incoming flits according to there transaction delay (Figure **??**). At point *t0* the complete flit is received and stored. But it can only partake in the arbitration when the annotated transaction delay has passed at *t1*. Thereby the execution order of the routers does not influence the outcome. In contrast, when directly applying the LT and AT coding styles, at least one respectively two dedicated threads are required per transaction. Performing transactions of all router ports through a single thread reduces the overhead and the memory footprint.

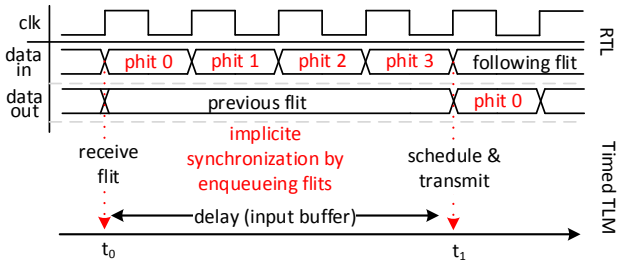

Fig. 3: Synchronization of neighboring routers via timed invalidation

The resulting router architecture still has a strong resemblance of the reference model (Figure **??**). But strictly applying object oriented programming techniques significantly increased the flexibility and maintainability. When optimizing a hardware layout with regard to propagation delay and occupied chip area it is not always possible to localize functionality to there designated modules. Unrestrained by these restriction the timed TL model encapsulates key functionality regarding the network characteristics (arbitration and QoS) into separated classes. The use of general interface definition of these classes enables independent development of alternative implementations. This increases the general design productivity as well as the exploration of different arbitration and QoS schemes.

As part of the TLM 2.0 library a set of socket implementations are provided to facilitate the connection between modules. Of these the multi pass through initiator and target sockets are used to create a full duplex connection between neighboring routers. Unlike typical RTL entity ports these multi sockets allow to construct N:M connections. By constructing each router with a target and initiator socket it becomes possible to configure any arbitrary topology beyond the scope of the reference model.

Call-back functions are bound to the target socket to facilitate the actual transaction. Here the TLM standard describes the use of blocking and nonblocking functions depending on the chosen coding style. But as stated in Section **??**, for networks the focus of timing accuracy shifts from individual transactions to the coordination between data streams. For the benefit of performance, our model implements nonblocking transport functions, as the timing behavior is modeled via buffer invalidation instead of explicitly blocking the initiator thread. Blocking the initiator would require individual threads for every router port and hence cause significant overhead. The functions are bound at configuration time, allowing for our model to differentiate between timed and untimed implementation at configuration time.

As shown in Figure **??**, incoming transactions are differentiated into flits and flow control signals named credits. Flits are stored into the input buffer and invalidated according to their transaction delay. Credits are registered by the credit counter representing a freed buffer location of a downstream router. The only active submodule is the arbiter. It is invoked when there are valid flits requesting to be forwarded. Fully contained in the arbiter are the scheduling and QoS modules which implement generic interfaces definitions. After a successful arbitration the corresponding flits are transmitted trough the switch fabric module accessing the initiator socket.

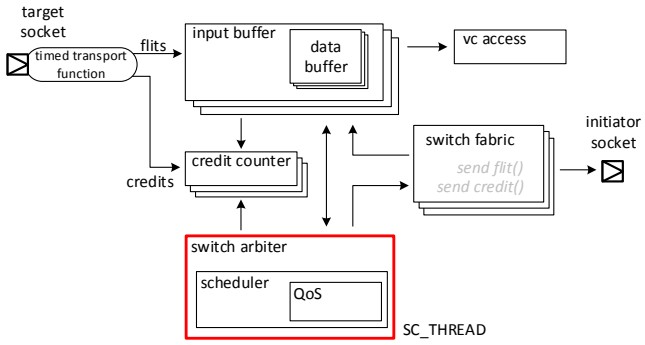

Fig. 4: Router architecture of the timed simulation model

C. Untimed Model

As stated above, for some use cases simulation performance is more desirable than the accurate reproduction of packet latencies. For example, extended hardware/software co-simulations as part of early software development benefit greatly as there is no direct interest in the subjacent architecture. Our untimed simulation model therefor assumes a congestion free transmission resulting in latency only depending

on the hop count. In addition temporal decoupling is used to perform all transactions from source to destination in a single simulation step. For this model the obligation to synchronize a package goes over to the receiver (if it is necessary).

The resulting architecture differs greatly to the timed model as flits are not stored or synchronized to competing data streams. By not storing the flits along their route the need for flow control is also omitted. The resulting router model only consists of the transport function and the switch fabric directly connected to the corresponding sockets.

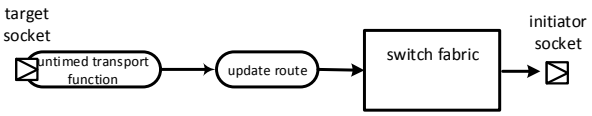

Fig. 5: Router architecture of the untimed simulation model

V. Evaluation

A. Test Configuration

The main interest of this work lies with simulation performance and timing accuracy. Therefore, a simple all-to-all traffic pattern is applied to the network. Traffic generators and receivers are connected to every router (Figure **??**), sequentially sending packets to all other receivers without any wait-states. The resulting high network load and frequent congestion are very demanding, minimal deviations from the reference behavior will quickly cause high latency and flit-sequence errors.

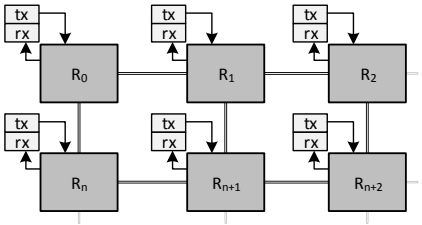

Fig. 6: 2D mesh NoC topology test configuration with traffic generators and receivers

The packet generators/receivers are implemented natively in VHDL and SystemC to allow fair performance comparisons of the abstraction levels. Questasim version 10.1d is used for the RTL simulations and OSCI SystemC 2.3.0 simulator for the TL models. Both were executed on a 3GHz Intel i7 64-Bit Windows 7 PC with 8 GB of RAM.

B. 8x8 NoC Measurements

To generate a high simulation effort, this configuration uses the biggest number of routers supported by the reference design. With 64 routers connected to an 8x8 two-dimensional mesh, traffic is generated from 10 up to 100 all-to-all iterations, consisting of one packet to each other router. At 5 flits per packet, this cumulates into the handling of 2.016.000 flits at 100 iterations. The resulting simulation execution-time is displayed logarithmically in Figure **??**. Herein, the untimed TLM model achieves a speed-up ranging between 1453 and 2031 compared to the RTL simulation. The higher-accuracy timed TL model still achieves a speed-up ranging from 162 to 175 compared to the RTL simulation.

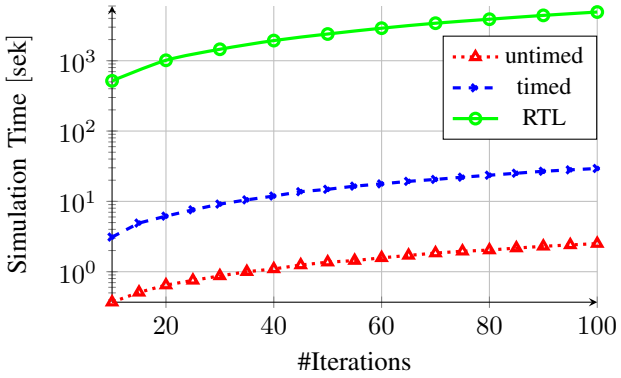

Fig. 7: 64 router NoC simulation speed (logarithmic) of timed, untimed and RTL models

Assessment of the timing accuracy is done by capturing the end-to-end latency of every flit. The comparison of the average latencies is shown in Figure **??**. By not modeling congestion, the untimed model assumes the best-case latency only depending on the average hop count. The average deviation of timed model is never greater than one clock cycle (the individual flit end-to-end latency error never exceeds three clock cycles), allowing precise behavioral analysis at significantly higher performance compared to the RTL model.

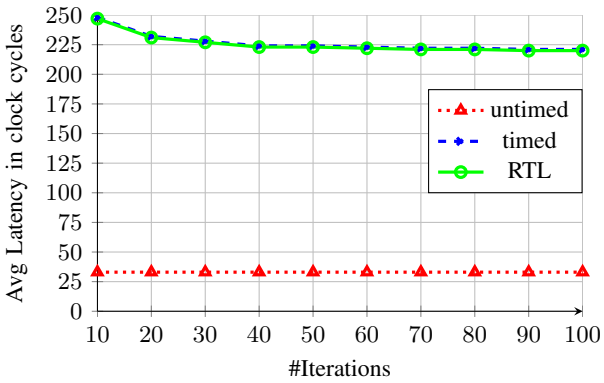

Fig. 8: Avg. flit latency in a 64 router NoC

C. Large NoC Simulations

Further exploration was done with varying network dimensions. Results are shown in Figure **??**. At constant 10 all-to-all-iterations the router count varies from 9 to 2500. Simulations of 50x50 networks causes a high number of individual threads to be handled by the SystemC simulation kernel. The timed

simulation model requires one thread per router and one thread per traffic generator. Scheduling up to 5000 threads in a single threaded simulation process results in numerous context switches and high run-time overhead. Future accelerating of large on-chip networks may be achieved by application of a parallel simulation kernels as presented in [**?**] and [**?**]. Minimizing the cost of synchronization will become a major goal when high accuracy is needed.

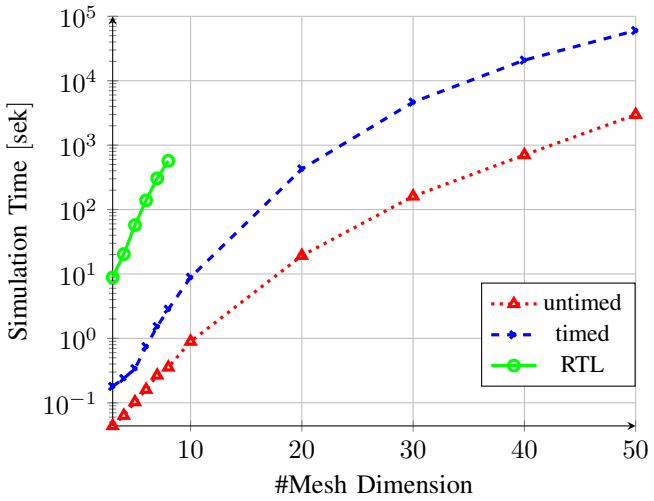

Fig. 9: 9 - 2500 router NoC simulation speed (logarithmic) of timed, untimed and RTL models (RTL model supports up to 8x8 routers)

VI. Conclusion

In this paper we demonstrate the performance and flexibility benefits of a raised modeling abstraction applied to a RTL NoC model. By translating and applying the TLM2.0 coding styles to a reference design we developed a simulation model with adjustable timing accuracy. Our timed simulation model was able to accurately reproduce the reference behavior while accelerating the simulation up to 175 times. Further acceleration was achieved at cost of accuracy. Assuming zero load conditions, our untimed simulation was executed up to 2031 times faster than the RTL model. In addition the design productivity was further raised by application of modern software engineering techniques. Allowing faster development and easier maintenance in the future.

Applying high performance simulation models will simplify the design of larger on chip networks. Moreover, supplying efficient virtual prototypes enables accelerated software development and verification.

VII. Acknowledgment

The research leading to these results has received funding from the ARTEMIS Joint Undertaking under grant agreement n 621429 and from the German Federal Ministry of Education and Research (BMBF), funding contract 01IS14002O.