Center for Embedded Computer Systems
University of California, Irvine

Using Result Oriented Modeling for Fast yet Accurate TLMs

Gunar Schirner, Rainer Dömer

Technical Report CECS-05-05
May 5, 2005

Center for Embedded Computer Systems
University of California, Irvine
Irvine, CA 92697-3425, USA
(949) 824-8059

hschirne@uci.edu, doemer@uci.edu
http://www.cecs.uci.edu/

Using Result Oriented Modeling for Fast yet Accurate TLMs

Gunar Schirner, Rainer Dömer

Technical Report CECS-05-05
May 5, 2005

Center for Embedded Computer Systems
University of California, Irvine
Irvine, CA 92697-3425, USA
(949) 824-8059

hschirne@uci.edu, doemer@uci.edu
http://www.cecs.uci.edu

Abstract

Communication modeling is a critical part during SoC design and exploration. In particular, it is needed for accurately predicting the timing behavior of the system. Fast simulation capabilities are a key in this environment, for coping with the complex design choices during the specification process. Recently, Transaction Level Models have been proposed to speedup communication simulation at the cost of accuracy.

This paper proposes a new modeling style: Result Oriented Modeling (ROM). Using ROM yields fast executing models, that still are 100% accurate. ROM utilizes the fact, that internal state changes of the model are not observable by the caller. Hence, it omits the internal states and optimistically predicts the end result. *Retroactively, the outcome is checked and if necessary, corrective measures are taken to maintain accuracy of the model.*

The paper reports how Result Oriented Modeling can be applied to Transaction Level Modeling for a communication system. It shows its application to two different bus systems: the AMBA AHB and the CAN bus. The initial results shown in this paper are very promising. Both implemented ROMs exhibit a performance in the same order of magnitude as the transaction level model, yet they retain the accuracy of the bus functional model. This clearly indicates that the proposed Result Oriented Modeling approach is a very effective modeling style for transaction level models.

Contents

List of Figures

List of Acronyms

AHB Advanced High-performance Bus. System bus definition within the AMBA 2.0 specification. Defines a high-performance bus including pipelined access, bursts, split and retry operations.

AMBA Advanced Microprocessor Bus Architecture. Bus system defined by ARM Technologies for system-on-chip architectures.

APB Advanced Peripheral Bus. Peripheral bus definition within the AMBA 2.0 specification. The bus is used for low power peripheral devices, with a simple interface logic.

ASB Advanced System Bus. System bus definition within the AMBA 2.0 specification. Defines a high-performance bus including pipelined access and bursts.

ATLM Arbitrated Transaction Level Model. A model of a system in which communication is described as transactions, abstract of pins and wires. In addition to what is provided by the TLM, it models arbitration on a bus transaction level.

BFM Bus Functional Model A wire accurate and cycle accurate model of a bus.

CAN Controller Area Network Serial communications protocol with a focus for automotive applications.

MAC Media Access Control. Layer within the OSI layering scheme.

OSI Open Systems Interconnection. An communication architecture model, described in seven layers, developed by the ISO for the interconnection of data communication systems.

SoC System-On-Chip. A highly integrated device implementing a complete computer system on a single chip.

TLM Transaction Level Model. A model of a system in which communication is described as transactions, abstract of pins and wires.

ROM Result Oriented Modeling A novel approach for fast and abstract modeling of a process with limited visibility to internal state changes.

Using Result Oriented Modeling for Fast yet Accurate TLMs

Gunar Schirner, Rainer Dömer

Center for Embedded Computer Systems
University of California, Irvine
Irvine, CA 92697-3425, USA

hschirne@uci.edu, doemer@uci.edu
http://www.cecs.uci.edu

Abstract

Communication modeling is a critical part during SoC design and exploration. In particular, it is needed for accurately predicting the timing behavior of the system. Fast simulation capabilities are a key in this environment, for coping with the complex design choices during the specification process. Recently, Transaction Level Models have been proposed to speedup communication simulation at the cost of accuracy.

This paper proposes a new modeling style: Result Oriented Modeling (ROM). Using ROM yields fast executing models, that still are 100% accurate. ROM utilizes the fact, that internal state changes of the model are not observable by the caller. Hence, it omits the internal states and optimistically predicts the end result*. Retroactively, the outcome is checked and if necessary, corrective measures are taken to maintain accuracy of the model.*

The paper reports how Result Oriented Modeling can be applied to Transaction Level Modeling for a communication system. It shows its application to two different bus systems: the AMBA AHB and the CAN bus. The initial results shown in this paper are very promising. Both implemented ROMs exhibit a performance in the same order of magnitude as the transaction level model, yet they retain the accuracy of the bus functional model. This clearly indicates that the proposed Result Oriented Modeling approach is a very effective modeling style for transaction level models.

1 Introduction

System-On-Chip (SoC) design faces a gap between the production capabilities and time-to-market pressures. The design space, to be explored during the SoC design, grows with the improvements in the production capabilities, while at the same time shorter product life cycles force an aggressive reduction of the time-to-market. Addressing this gap has been the aim of recent research work. As one approach, abstract models have been introduced to tackle the design complexity.

Fast simulation capabilities are required for coping with the immense design space that is to be explored; these are especially needed during early stages of the design. This need has pushed the development of transaction level models (TLM) [6], which are abstract models that execute dramatically faster than synthesizable, bit-accurate models.

Transaction level modeling, however has usually the drawback of decreased accuracy. This paper introduces a novel modeling technique, which aims to deliver the speedup of transaction level models, yet it retains the accuracy of a bus functional model. The apparently incompatible goals are achieved by using the following key points:

1. Bus communication is simulated at the level of user transactions. That means a block of contigues bytes in memory is transfered from one bus node to another bus node.
2. The computational behaviors that are using the bus communication can only observe the effects

of the bus communication at the boundaries of the user transaction.
3. All events and data transfers which are necessary to facilitate the transfer of the user transaction can be freely rearranged and regrouped.
4. If the transaction internal events are rearranged so that there are fewer interactions between bus nodes (communication partners), a performance gain will be achieved.

By using the ROM that provides 100% accuracy at a high speed, the model user is relieved from the traditional speed / accuracy tradeoff. This allows the designer to focus on the design space exploration instead of the model selection.

This paper will first describe the general concept of the Result Oriented Modeling independent of the its application to communication modeling. It will then show an application of the Result Oriented Modeling for modeling an AMBA AHB bus. It will point out the limitations of the current way of communication modeling, show the application of the Result Oriented Modeling (ROM) approach and outline initial results. The paper then continues with a second application of the ROM concept to communication modeling of a serial bus, the Controller Area Network. The paper will conclude by analyzing the experimental results of both applications and will give an outlook for the future research direction.

1.1 Related Work

System level modeling has become an important issue, as a means to improve the SoC design process. Languages for capturing such models have been developed (e.g. SystemC [6], SpecC [5]). Capturing and designing communication systems using TLMs [6] has received attention.

Sgroi et al. [9] address the SoC communication with an Network-on-Chip approach. They propose partitioning the communication into layers following the OSI structure.

Siegmund and Müller [13] describe with SystemC^{SV} an extension to SystemC, and propose SoC modeling three different levels of abstraction: the physical description at RTL level, a more abstract model for individual messages, and a most abstract level that deals with transactions.

Coppola et al. [3] propose an abstract communication modeling, present the IPSIM framework and show its efficient simulation, however no accuracy analysis.

Gerstlauer et al. [4] describe a layered approach and propose models that implement an increasing number of OSI [7] layers. They have shown speedup of at most 100x, however the accuracy analysis is very limited.

In previous work we have modeled the Controller Area Network (CAN) [11] and the Advanced Microprocessor Bus Architecture (AMBA) [12] using an OSI layer based approach. In this paper we will significantly improve on these results by using the ROM idea outlined in the 4 bullet points above.

Pasricha et al. cover in [14] a similar approach of a transaction-based abstraction level. The paper introduces the concept of a model that is Cycle Count Accurate at Transaction Boundaries (CCATB). It too takes advantage of the limited observability within a transaction to increase the performance. However, only a limited performance gain was achieved. While maintaining the bus cycle accuracy, the CCATB approach achieved a 55% speedup over the bus functional model. This limited result may be due to the utilized instruction set simulator.

2 The Result Oriented Modeling Concept

The Result Oriented Modeling (ROM) approach is a general concept for an abstract and yet accurate modeling of a process. Its underlying assumption is the limited observability of the internal changes of the modeled process. It is often not required, that the intermediate results of the process are visible to the user of that process. In this respect ROM is similar to a ”black box” approach. The main goal of the ROM process modeling is to produce the *end result* of that model, not the intermediate status changes.

Limited observability of the intermediate status changes, gives ROM the opportunity for optimization. It can eliminate the explicit modeling of intermediate state changes. Instead, ROM can utilize an optimistic approach, that predicts the outcome (e.g. termination time and final state) of the process at the time

the process is started.

Throughout the simulated running time of the process a *disturbing influence* may change the system state, so that the initial predicted results are no longer accurate. Therefore, ROM checks at the end of the predicted process time whether such an *disturbing influence* has occurred. In that case the ROM retroactively adjusts to the new conditions and takes corrective measures. In other words, the mistake of an overly optimistic initial prediction is fixed at the end.

Optimistic prediction of the *end result* and retroactive correcting the result reduces the amount of computation and thus increases the execution performance, if internal states can be skipped. This two step approach is in contrast to the more traditional abstract modeling approach of reaching the *end result* through a set of incremental internal state changes. The traditional approach takes the *disturbing influence* incrementally into account and adjusts the intermediate state changes accordingly. ROM, on the other hand, records the *disturbing influence* over the predicted running time and makes an adjustment at the end.

Generally speaking, the ROM approach can be characterized by the following items:

1. The model user does not need to observe process internal state changes.
2. ROM does not model internal state changes, instead it optimistically predicts the *end result* with the current system knowledge
3. Throughout the predicted run time of the process a *disturbing influence* may change the system state.
4. At the end of the process, ROM checks if the optimistic assumptions still hold true and takes corrective measures if necessary.

Repeating the ”black box” comparison, ROM is a ”black box” approach that explicitly includes interaction with other ”black box” instances (as *disturbing influence*) and takes corrective measures in case the interaction is not as predicted.

To contrast ROM to a traditional Transaction Level Model (TLM), traditional abstract modeling achieves speedup by modeling the internal state changes at a more coarse granularity and reaches the final result through an incremental approach taking into account any disturbing influence at each step. ROM on the other hand, optimistically predicts the *end result*, records the disturbing influence and only makes a correction in the end when it deems necessary.

Figure 1 illustrates the ROM approach using an example of predicting the arrival time of an airplane. The real process (a) exhibits continuous changes to the airspeed dependent on the disturbing influence wind. The traditional abstract modeling approach (b) approximates the result by incrementally calculating the air speed in dependence of the wind in (coarse) discrete time steps. The ROM approach (c), on the other hand, does not model intermediate airplane speed. Instead it makes one initial optimistic prediction about the arrival time and corrects its prediction retroactively for the average wind condition.

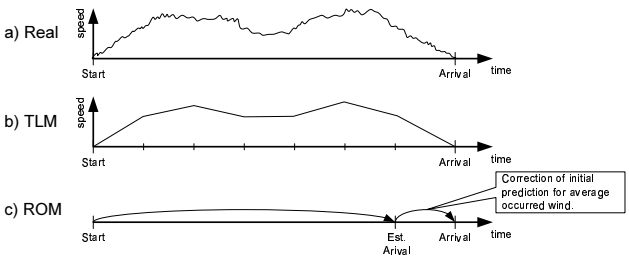

Figure 1: Illustration of different models for predicting an airplane arrival time.

The next section describes how the up to now generally described ROM concept can be applied to modeling of a communication system. It uses the example of an AMBA AHB, which is introduced first. It then describes the current way of modeling the AHB so that the reader has a frame of reference. After that, the application of ROM is described and an initial model implementation is analyzed.

3 Result Oriented Modeling applied to AMBA AHB

3.1 Introduction to the AMBA Bus

ARM defined with the Advanced Microprocessor Bus Architecture (AMBA) [2] a widely used on-chip bus system standard. It contains a group of busses, which are used hierarchically as shown in Figure 2. This paper focuses on the Advanced High-performance Bus

(AHB), a system bus designed for connecting high-speed components including ARM processors.

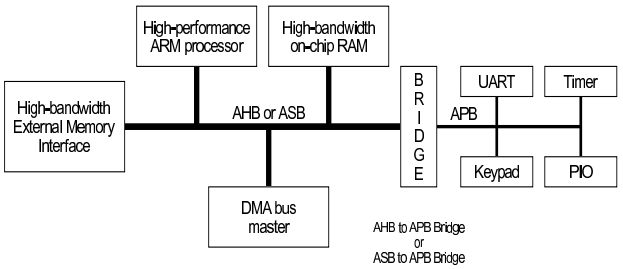

Figure 2: AMBA bus architecture
AMBA bus architecture (Source [2]).

The AHB is a multi-master bus that operates on a single clock edge. High performance is achieved by a pipelined operation that overlaps arbitration, address, and data phases, and by the usage of burst transfers. Split and retry transfers allow the slave to free the bus if the requested data is temporary unavailable. The AHB also employs a multiplexed interconnection scheme to avoid tri-state drivers.

3.2 Existing Modeling Approach

The existing modeling approach, as described in [12], uses a layered architecture to cope with the communication complexity. It follows the ISO OSI reference model [7]. In that respect the AHB specification falls within the second layer, the data link layer. For modeling of the AHB, the media access control (MAC) and the protocol sublayer are considered, as well as the physical layer.

Important for this discussion is the granularity of data handling in each of the layers. The *media access layer* provides a transmission service for a contiguous block of bytes, called a *user transaction*. This layer divides the arbitrary sized user transaction into smaller bus transactions observing the bus addressing rules and transfers these byte blocks using the protocol layer. The *protocol layer* transfers data as *bus transactions*, which are bus primitives (e.g. bytes, words, or 4 word burst). It uses the *physical layer* services, which provide a *bus cycle* access to sample and drive individual bus wires.

Figure 3 shows how the above defined data granularity levels can be analyzed with respect to time. A user transaction is successively split into the smaller elements: bus transactions and finally bus cycles.

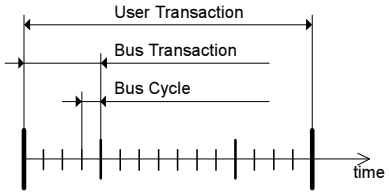

Figure 3: User Transaction Decomposition.

In the current approach [12], the abstraction is correlated to the number of implemented layers and the granularity of arbitration handling. The most abstract model, the TLM implements only the media access layer and handles arbitration at the coarse level of user transactions. [12] has defined three major models.

The *TLM* is the most abstract model; it only implements the media access layer. The user data, handled at the user transaction granularity, is transferred regardless of its size in one chunk using a single *memcpy*. Timing is simulated by a single *waitfor* statement, covering the whole user transaction. Arbitration is not modeled. Instead, concurrent access is resolved using a semaphore once per user transaction. Due to using a semaphore, the contention resolution depends on the simulation environment.

The *Arbitrated Transaction Level Model (ATLM)* simulates the bus access with a bus transaction granularity (AHB bus primitives), at the protocol layer level. It uses the MAC layer implementation of the Bus Functional Model (BFM) to split user transactions into bus transactions. The ATLM accurately models priority based arbitration, however only once for each bus transaction. This model is not pin accurate and not cycle accurate in all cases.

The *BFM* is a synthesizable, bus cycle accurate and pin accurate bus model. It implements all layers down to the physical layer and covers all timing and functional properties of the bus definition. It handles arbitration per bus transaction and verifies the bus grant on each cycle of a burst. We implemented additional active components, such as multiplexers, an arbiter and an address generator, for correctly modeling the bus architecture.

Figure 4 summarizes the described models. It shows for each model the lowest layer that it imple-

ments and the granularity at which it handles data and arbitration. The user transaction decomposition figure is superimposed to visualize the granularity.

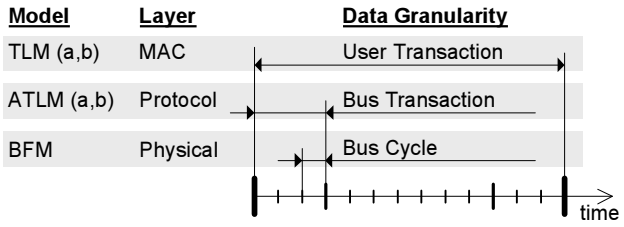

Figure 4: Model Summary.

To illustrate the limitations of the current approach, let us consider one transfer mode of the AHB bus: an unlocked burst transfer. In a burst transfer multiple data words are transferred over the bus as one block of data. An unlocked burst transfer may be preempted by a higher priority master. Hence, the active master has to check the arbitration for every bus cycle (beat). In case of a preemption, the preempted master has to arbitrate again for the bus and subsequently resume the preempted transfer.

Figure 5 shows in which granularity the arbitration checking (symbolized by a check mark) is done in the three models. An user transaction of 16 words is transfered in two 8-beat bursts. The BFM performs a full arbitration at the beginning of each bus transaction (each 8-beat burst) and also verifies the arbitration at each bus cycle. The ATLM checks arbitration only at the bus transaction boundary. The TLM performs the least amount of checking, and only arbitrates on the user transaction boundary.

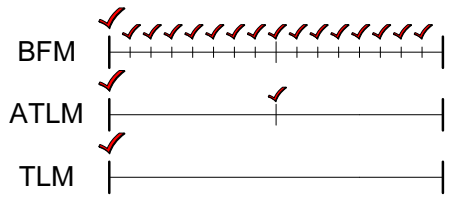

Figure 5: Arbitration check points when transferring two 8 beat bursts.

The amount of arbitration check points directly translates to the performance of the model. Implementing all required arbitration check points is the slowest, but delivers an accurate time prediction. The other extreme, as implemented by the TLM, is to implement the fewest arbitration check points, which yields the highest performance but results in the worst accuracy. The ROM, as it will be described in the next section, does not exhibit the correlation between granularity and abstraction. It avoids unnecessary arbitration check points. As a result, it reaches the BFM accuracy of 100% and near TLM performance.

3.3 Result Oriented Modeling Approach

3.3.1 Assumptions

The Result Oriented Modeling approach makes use of the assumption that the modeled applications do not have visibility of the actual transfer internals. It is assumed that the communication is separated from the computation. In such a scenario, the application is only aware of the timing at the boundaries of the user transaction. All activities of the bus model, that are required within a user transaction are hidden from the computation code. As such, the application is not aware that e.g. the user transaction is split into bus transactions, and neither that there is arbitration required.

With these assumptions, only the timing at the boundaries of the user transaction are of importance for the application timing. This limited observability allows ROM to rearrange and/or omit internal events of the data transfer in order to eliminate unnecessary context switches in the simulation.

3.3.2 Modeling

The ROM approach is based on the TLM idea; it avoids using signals and individual wires. Instead, it models the transfer of a complete user transaction with a single *memcpy*. The main speed contribution is replacing the multiple wait statements, as done in the bus functional model for checking the arbitration on each bus cycle, with one single wait statement. Reducing the number of wait statements is the biggest contributor to an increased execution performance. It avoids running the scheduling algorithm of the simulation engine and thus also reduces the number of context switches.

The ROM implements an optimistic approach. When master a requests a user transaction transfer,

the earliest finish time for this user transaction is calculated and the masters execution thread is delayed until this time. This time calculation takes the current status of the bus into account, e.g. by increasing the wait time for the completion of a currently running transaction. After the earliest termination time has passed, the ROM verifies whether the predicted transaction finish time is still accurate. Should that be the case, the transaction is finished. Note, that this scenario marks the best case. The ROM uses only a single wait statement - same as the TLM - and it hence shows the same execution performance as the TLM. Figure 6 depicts the optimal scenario for the transfer of 16 words. Here the ROM has reduced the number of wait statements from 16 to 1.

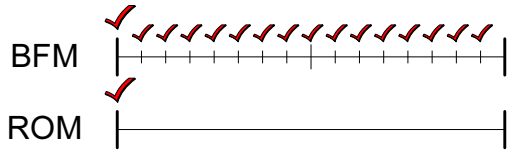

Figure 6: Arbitration Check Points BFM vs. ROM optimum case.

With a *disturbing influence* of a higher priority master accessing the bus, the initial finish time prediction may turn out to be inaccurate. Then, the ROM detects this at the predicted finish time and calculates the next earliest transaction finish time for transferring the remaining part of the transaction. Then, the master is delayed for the updated transaction finish time and the checking process subsequently repeats. Note that an optimistic prediction is necessary for any corrections. Assuming a pessimistic prediction (i.e. predicting a too long transfer time), the correction would need to go back in time, which obviously is not possible. Figure 7 and 8 show an example of a burst preemption.

Figure 7 shows the bus functional model behavior.

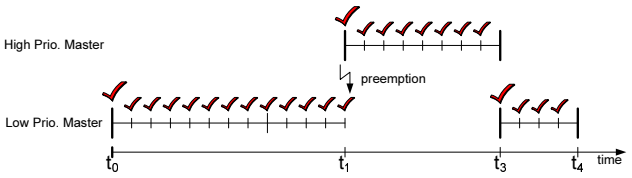

Figure 7: BFM preemption modeling.

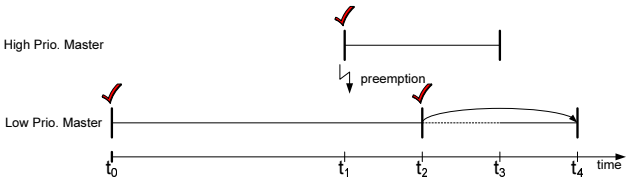

Figure 8: ROM preemption modeling.

	BFM	**ROM**
Arbitration Checks	25	5
Wait Statements	24	3

Table 1: Complexity comparison BFM v.s. ROM.

The burst started at t_0 is preempted by a higher priority master at t_1. The transfer of the higher priority master is finished at t_3 where the preempted burst of the low priority master is resumed. The BFM checks the arbitration 25 times, both masters are modeled with 24 wait statements in total.

The ROM, as shown in Figure 8, requires only 5 arbitration checks and a total of 3 wait statements. At t_0, the low priority transfer is estimated to finish at t_2, hence the low priority master is delayed until that time. At t_1 however, a the high priority master preempts the transfer and predicts a finish time for its own transfer of t_3. At t_2 the low priority master is scheduled again, it detects that its transfer was preempted. Hence, the low priority master adjusts its own waiting time. The adjusted transaction finish time is computed as t_4, which is composed of waiting time until the high priority master finishes (until t_3) and the time for the remaining transfer. The high priority master wakes up at t_3 and terminates the transaction, since its transfer did not get preempted. At t_4 the low priority master wakes up and terminates the transaction, since the resumed transfer did not get preempted.

Table 1 summarizes the complexity in terms of arbitration checks and wait statements for the example transfer. The ROM only performs 20% of the check and wait operations.

3.3.3 Challenges

Already writing a Transaction Level Model is challenging to the model designer. The bus standard description (e.g. [1]) describes the interaction between

the bus nodes on a bus cycle level using individual wires. The TLM on the other hand has to abstract this information and implement the timing implicitly. An increased challenge is to eliminate a separate flow of execution for arbitration. Then, concurrent access to the bus has to be handled in each master and the arbitration has to be implemented implicitly.

The ROM approach poses additional challenges on top of those of the TLM. It requires that the bus model has to keep track of outstanding transactions, it has to be able to backtrack decisions (i.e. if they where overly optimistic) and recover partial transfers.

3.4 Experimental Results

In order to estimate the benefits of the proposed approach we have extended the performance analysis of the AMBA AHB models, as described in [12], by one model that uses the ROM approach. To minimize the implementation effort, the model includes only the prediction of the earliest transaction finish time. It does not yet implement the resolution algorithm to back trace a too optimistic prediction (this remains for future work).

We have examined the simulation performance of each model in a scenario with one master and one slave. User transactions are transferred repeatedly, without any delay in between. We have measured the simulation time (also referred to as real time or wall clock time) for all transfers and computed the simulated bandwidth. All tests have been performed on a Pentium 4 at 2.8 GHz.

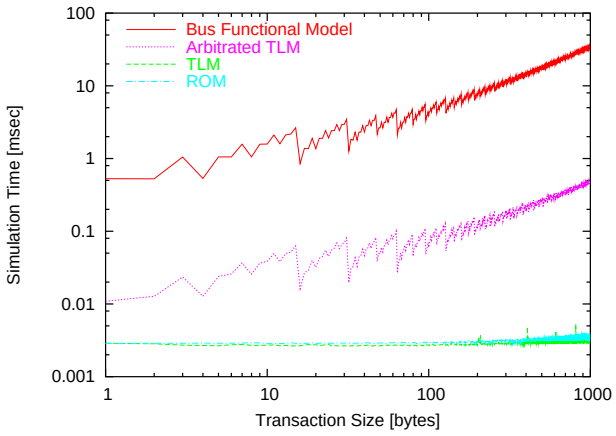

Figure 9: AMBA Model Transfer time.

As Figure 9 shows, the ROM model is as fast as the TLM. This result marks the best possible performance achievable by the ROM. Since only a single master is connected to the simulated bus, there is no need for revising a prediction of a transaction finish time. Hence, in this scenario the ROM executes the same operations as the TLM and shows identical high performance.

For future measurements, we will include the influence of at least one competing master. Then, the ROM execution time will, depending on the amount of bus contention, be slower than the TLM. However, it is still predicted to be much faster than the ATLM, due to fewer wait statements.

4 Result Oriented Modeling applied to CAN

To get a broader view of the Result Oriented Modeling potential, we have also applied this approach to a completely different bus system. While the AMBA AHB is an on chip parallel bus system, the second bus system, the CAN is an off chip serial bus designed for automotive applications. The following section gives a short introduction of the CAN bus (quoted from [11]).

4.1 Introduction to CAN the Bus

The Controller Area Network (CAN) is a serial communications protocol, introduced by the Robert Bosch GmbH [10], that was designed with a focus for automotive applications. It supports ”intelligent” networking devices as well as sensors and actuators on the same bus system.

The CAN serial bus is a multi master broadcast bus. Messages carrying 0 to 8 Bytes of data are received by all bus nodes and distinguished by the message identifier. Each bus node decides using local rules if it should process the received message. This principle of using message identifiers is in contrast to standard bus systems, where an address of a transfer uniquely selects a communication partner.

The message identifier serves also as a message priority. Should multiple masters attempt to send a message at the same time, the collision free CSMA/CA

arbitration will result that the sender with the highest priority will succeed.

The CAN bus defines two bus states. One is recessive (1) and the other dominant (0). A CAN data frame has the basic format shown in Figure 10. After transmitting the start of frame bit, the message identifier is transmitted with the most significant bit first. During transmission, each sender validates that the sent bit is equal to the received bit. If two senders start transmission of a frame at the same time, they both will transmit their message id at the same time. The sender with the higher message identifier (the lower priority) will detect, that although it has send a recessive bus level (1), it received a dominant bus level (0) and it will back off from transmission. Sending of the higher priority message continues, the message remains intact.

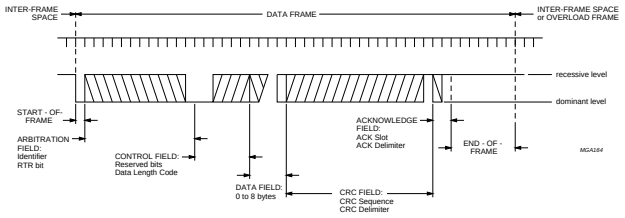

Figure 10: CAN Data Frame
CAN Data Frame (Source [8]) .

In order to ensure correctness of the received data, each CAN message includes a 15-bit CRC. In case the transmitted CRC does not match the computed CRC a retransmission of the frame is triggered. The protocol furthermore defines elaborate error detection and error confinement rules, to protect the bus from faulty bus nodes.

The CAN serial protocol operates without a centralized clock. Each bus node synchronizes on the bit stream of the sender. The Non-Return-to-Zero coding is extended by a bit stuffing rule, which guarantees that there are sufficient edges in the bit stream for this synchronization. After transmitting 5 bits of equal polarity, a bit of opposite polarity is introduced.

In summary, the following five properties are of interest for choosing the models of abstraction:

- Serial protocol
- Bit synchronization
- Error detection and confinement
- Bit error detection using a 15 Bit CRC
- Bit stuffing
- Arbitration, bus access controlled by CSMA/CA

4.2 Results for the CAN Bus

As with the previous introduced AHB, we have implemented an initial version of the ROM model for CAN. It also is based on an existing TLM [11]. However, in order to guarantee correct timing, it has to perform more work than the TLM. Due to the bit stuffing rule, the number of bits transferred depends on the message content. Therefore, the ROM has to perform a bit inspection of the message to calculate the CRC and insert the stuffing bits. The initial ROM version does not include the algorithm for retroactively adjusting the timing prediction.

Figure 11 shows the performance of the ROM model in the context of previous measurements [11]. Two nodes are connected to the simulated CAN bus, one node sends user transactions of increasing size, the other node receives them.

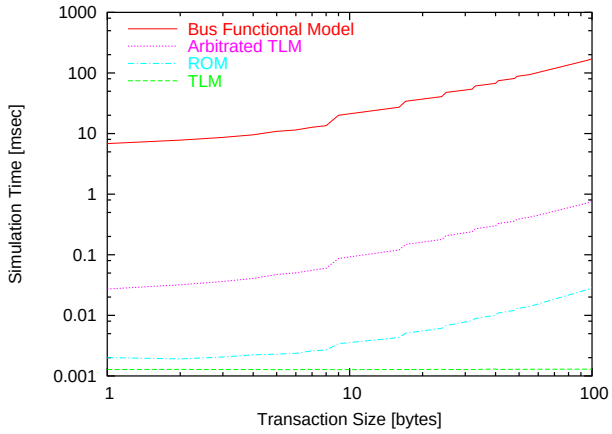

Figure 11: CAN Model Transfer Time

The results as shown in Figure 11 are very promising. The ROM is an order of magnitude faster than the ATLM. For small messages, the ROM is in the same order of magnitude as the TLM. However, it is slower for messages bigger than 8 bytes due to the additional effort of bit inspecting the message (CRC and bit stuffing calculation). Table 2 shows the results numerically for transferring a 16 byte message. For this data point, the ROM is 30 times faster than the ATLM and only 3 times slower than the TLM.

Feature	BF	ATLM	ROM	TLM
Sim. Time [ms]	27.3	0.120	0.0043	0.0013
Sim. Bandwidth [MBytes/s]	0.0006	0.13	3.534	11.9
BFM Speedup	1	227	6313	21330
Rel. Speedup	1	227	27.8	3.4

Table 2: Transfer size: 16

5 Conclusions

This paper has introduced a new modeling style, Result Oriented Modeling, which produces results as needed by the calling application and hides internal state changes. Hiding the internal state changes is used as an advantage to optimistically predict the outcome, and retroactively correcting it in case there was a disturbing influence.

The paper has reported on an application of the ROM concept to create a fast yet accurate Transaction Level Model for modeling communication systems. Two initial implementations of an ROM style TLM (AMBA Advanced High-performance Bus (AHB) and CAN) have been described and measurements have been presented that estimate significant benefits of the ROM approach.

The initial measurements have shown very promising results. For the AMBA AHB, the ROM was as fast as the TLM, and for the CAN, the ROM was in the same order of magnitude for small messages. These results lead us to the conclusion, that the novel Result Oriented Modeling approach is very promising. In future work, we will complete the ROM implementation and quantitatively analyze the benefits including the necessary corrective measures.

References

[1] Advanced RISC Machines Ltd (ARM). AMBA Home Page. `www.arm.com/products/solutions/AMBAHomePage.html`.

[2] Advanced RISC Machines Ltd (ARM). AMBA Specification (Rev. 2.0), ARM IHI 0011A. `www.arm.com/products/solutions/AMBA_Spec.html`.

[3] Marcello Coppola, Stephane Curaba, Miltos Grammatikakis, and Giuseppe Maruccia. IPSIM: SystemC 3.0 enhancements for communication refinement. In *Proceedings of the Design, Automation and Test in Europe (DATE) Conference*, Munich, Germany, March 2003.

[4] A. Gerstlauer; D. Shin; R. Doemer; D. Gajski. System-Level Communication Modeling for Network-on-Chip Synthesis. In *Asia and South Pacific Design Automation Conference*, Shanghai, China, January 2005.

[5] Daniel D. Gajski, Jianwen Zhu, Rainer Dömer, Andreas Gerstlauer, and Shuqing Zhao. *SpecC: Specification Language and Design Methodology*. Kluwer Academic Publishers, 2000.

[6] Thorsten Grötker, Stan Liao, Grant Martin, and Stuart Swan. *System Design with SystemC*. Kluwer Academic Publishers, 2002.

[7] Internation Organization for Standardization (ISO). *Reference Model of Open System Interconnection (OSI)*, second edition, 1994. ISO/IEC 7498 Standard.

[8] Philips. P8xc592: 8-bit microcontroller with on-chip can. `www.semiconductors.philips.com`, 1996.

[9] M. Sgroi; M. Sheets; M. Mihal; K. Keutzer; S. Malik; J. Rabaey; and A. Sangiovanni-Vincentelli. Addressing the system-on-a-chip interconnect woes through communication based design. In *Proceedings of the Design Automation Conference*, June 2001.

[10] Robert Bosch GmbH. *CAN Specification*, 2.0 edition, 1991. `http://www.can.bosch.com/`.

[11] G. Schirner and R. Dömer. Abstract Communication Modeling: A Case Study Using the CAN Automotive Bus. In A. Rettberg, M. Zanella, and F. Rammig, editors, *From Specification to Embedded Systems Application*, Manaus, Brazil, August 2005. Springer.

[12] G. Schirner and R. Dömer. Quantitative Analysis of Transaction Level Models for the AMBA Bus. In *Proceedings of the Design, Automation and Test in Europe (DATE) Conference*, Munich, Germany, March 2006.

[13] Robert Siegmund and Dietmar Müller. SystemC^{SV}: An extension of SystemC for mixed multi-level communication modeling and interface-based system design. In *Proceedings of the Design, Automation and Test in Europe (DATE) Conference*, Munich, Germany, March 2001.

[14] Mohamed Ben-Romdhane Sudeep Pasricha, Nikil Dutt. Fast exploration of bus-based on-chip communication architectures. In *CODES and ISSS*, Stockholm, Sweden, September 2004.