Fast and Accurate Transaction Level Models using Result Oriented Modeling

Gunar Schirner and Rainer Dömer
Center for Embedded Computer Systems
University of California, Irvine, USA
{hschirne,doemer}@uci.edu

ABSTRACT

Efficient communication modeling is a critical task in SoC design and exploration. In particular, fast and accurate communication is needed to predict the performance of a system. Recently, Transaction Level Modeling (TLM) is used to speedup communication simulation at the cost of accuracy.

This paper proposes a novel modeling technique called Result Oriented Modeling (ROM) which removes the accuracy drawback of TLM. Using ROM, models yield the same speed as their TLM counterparts, yet still are 100% accurate in timing. ROM utilizes the fact that internal states in the communication channel are not observable by the caller. Hence, ROM omits the internal states entirely and optimistically predicts the *end result*. Retroactively, the outcome is checked and, if necessary, corrective measures are taken to maintain the accuracy of the model.

In this paper, we apply ROM to the AMBA AHB bus architecture. Our experimental results show that ROM exhibits the same high simulation performance as traditional TLM, yet it retains the same accuracy as the bus functional model. Thus, the proposed ROM approach eliminates the speed/accuracy tradeoff exhibited by traditional TLM.

1. INTRODUCTION

System-On-Chip (SoC) design faces a gap between the production capabilities and time-to-market pressures. The design complexity grows with the improvements in 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 system-level research. As one solution, abstract models have been used.

Fast simulation is required especially during the early stages of the design process. This need has pushed Transaction Level Modeling (TLM) [5] which utilizes abstract models that execute dramatically faster than synthesizable, bit-accurate models. TLM, however, usually comes at the price of significantly reduced accuracy.

In this paper, we introduce a novel modeling technique, called Result Oriented Modeling (ROM), which delivers the same speed as TLM. At the same time, ROM retains 100% accuracy in timing. These apparently incompatible goals are achieved using the following key ideas:

1. As with TLM, bus communication is simulated at the level of user transactions. For every transaction, a contiguous block of data is transferred from one bus node to the other by use of a single *memcopy* operation.
2. The communicating parties observe the effects of the bus transfer only at the boundaries of the transaction.
3. Events and context switches needed to facilitate the transfer are hidden and therefore can be freely rearranged.
4. If the arbitration checks and internal events are rearranged so that there are fewer context switches, a performance gain will be achieved.

Using these ideas, we will describe ROM for the AMBA AHB architecture [1], providing 100% accuracy at highest simulation speed. As a result, the system designer is relieved from the traditional speed/accuracy tradeoff and can focus on the design space exploration instead of model selection.

1.1 Related Work

System level modeling has become an important research area that aims to improve the SoC design process. Languages for capturing SoC models have been developed, e.g. SystemC [5] and SpecC [3]. Capturing and designing communication architectures using TLM [5] has received much attention.

Sgroi et al. [10] address the SoC communication with a Network-on-Chip approach. Here, communication is partitioned into layers following the OSI structure.

Siegmund and Müller [11] describe with SystemC^{SV} an extension to SystemC and propose SoC modeling at three different levels of abstraction: physical description at RTL, a more abstract model for individual messages, and a most abstract model utilizing transactions.

Coppola et al. [2] also propose abstract communication modeling. They present the IPSIM framework and show its efficient simulation.

Gerstlauer et al. [4] describe a layered approach and propose models that implement an increasing number of OSI [6] layers. Simulation speedup up to 100x is shown, but the accuracy analysis is limited and shows significant drawbacks.

Pasricha et al. [8] describe a similar approach using transaction-based abstraction. The paper introduces the concept of a model that is cycle count accurate at transaction boundaries (CCATB). This also takes advantage of the limited observability of a transaction to increase simulation performance. However, only a very limited speedup of 55% over the bus functional model is achieved[1].

In Section 2, we will introduce the general concept of Result Oriented Modeling, independent of its application to communication modeling. We will then show in Section 3 the application of ROM to the AMBA AHB bus architecture. We will also analyze the limitations of traditional communication modeling and show the advantages of the ROM approach. Section 4 provides experimental results that clearly support the claimed benefits of ROM. Finally, Section 5 concludes this paper with a summary and directions of future work.

2. RESULT ORIENTED MODELING

Result Oriented Modeling (ROM) is a general concept for abstract and yet accurate modeling of a process. As such, ROM is similar to the "black box" concept.

2.1 Black Box Concept

The underlying assumption of ROM is the limited observability of internal state changes of the modeled process. It is not necessary to show intermediate results of the process to the user, as in a "black box" approach. The only goal of Result Oriented Modeling (ROM) is to produce the *end result* of the process, not any intermediate states.

Hiding of intermediate states gives ROM the opportunity for optimization. Often, intermediate states can be entirely eliminated. Instead, ROM can utilize an optimistic approach that predicts the outcome (e.g. termination time and final state) of the process already at the time the process is started.

2.2 Corrective Measures

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

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

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

1. The user does not need to observe internal states.
2. ROM does not model internal state changes. Instead, it optimistically predicts the *end result* using available system information at the beginning.
3. During the predicted runtime of the process, a *disturbing influence* may change the system state.
4. At the end, ROM checks if the optimistic assumptions still hold true, and takes corrective measures otherwise.

[1]Our results show a speedup of three orders of magnitude.

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

2.3 Example

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-grain) discrete time steps. The ROM approach (c), on the other hand, does not model the intermediate airplane speed. Instead, it makes one initial optimistic prediction about the arrival time, and finally corrects its prediction retroactively for the average wind condition.

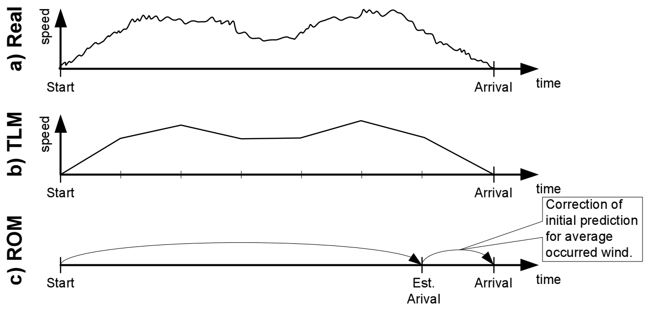

Figure 1: ROM predicting an airplane arrival time.

3. AMBA BUS MODELING USING ROM

We will now describe how the general ROM concept can be applied to modeling of a communication system. We will use the example of an AMBA AHB, which is introduced first. We then describe a set of layer-based AHB models as reference. Finally, we will apply the ROM approach to the AHB and analyze its benefits.

3.1 Introduction to the AMBA Bus

ARM has defined a widely used on-chip bus standard with the Advanced Microprocessor Bus Architecture (AMBA) [1] which contains a hierarchy of busses as shown in Figure 2. For this paper, we will focus on the Advanced High-performance Bus (AHB), a system bus designed for connecting high-speed components including ARM processors.

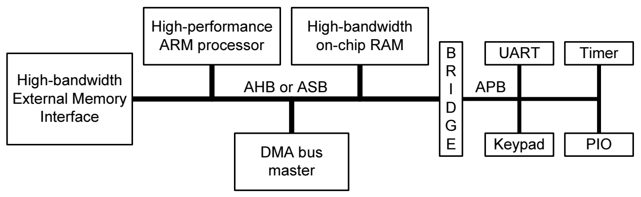
High-performance ARM processor
High-bandwidth on-chip RAM
High-bandwidth External Memory Interface
AHB or ASB
DMA bus master
BRIDGE
UART
Timer
APB
Keypad
PIO

Figure 2: AMBA bus architecture [1].

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 Layer-based Modeling

Following the ISO OSI reference model [6], we can model the AHB using a layered architecture [9]. The AHB specification then falls into the second layer, the data link layer. For modeling, we consider the media access control (MAC) and the protocol sublayer, 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 arbitrarily 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 services of the *physical layer* which provide a *bus cycle* access to sample and drive individual bus wires.

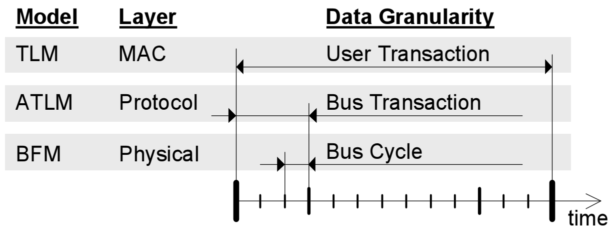

Figure 3: Layer-based Bus Modeling.

Figure 3 shows the data granularity at each layer with respect to time. A user transaction is successively split into smaller units: bus transactions and bus cycles.

Following this layering, we can define three models which we will refer to as TLM, ATLM, and BFM.

3.2.1 Transaction Level Model (TLM)

The TLM[2] is the most abstract model, implementing only the media access layer. Data is handled at the user transaction granularity and is transferred regardless of its size in one chunk using a single *memcpy*. Timing is simulated as a single *wait-for-time* statement, covering the entire user transaction. Arbitration is abstracted to a semaphore used once per user transaction.

3.2.2 Arbitrated Transaction Level Model (ATLM)

The ATLM models a bus access with AHB bus primitives at the protocol level. It uses the MAC layer implementation of the bus functional model 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 in all cases cycle-accurate.

[2]Note that TLM is not clearly defined in the literature. For this paper, we will use TLM as the name of the model at the granularity of an entire user transaction.

3.2.3 Bus Functional Model (BFM)

The BFM is a synthesizable, cycle- 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. The BFM handles arbitration per bus transaction and verifies the bus grant on each cycle of a burst. We have implemented additional active components, such as multiplexers, an arbiter and an address generator to accurately model the bus architecture.

3.2.4 Limitations of layer-based models

To illustrate the limitations of the layer-based approach, let us consider 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 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.

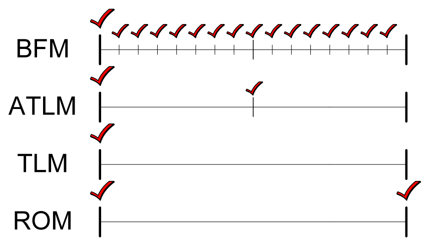

Figure 4: Arbitration check points when transferring two 8-beat bursts.

Figure 4 shows the arbitration check points as check marks for the three models. A user transaction of 16 words is transfered in two 8-beat bursts. The BFM performs full arbitration at the beginning of each bus transaction and also verifies the arbitration at each cycle. The ATLM checks arbitration only at the bus transaction boundaries. The TLM performs the least amount of checking. It arbitrates only at the beginning of a user transaction.

As we will see below, the ROM proposed in this paper performs two arbitration checks, at the beginning and at the end of a user transaction, i.e. one more than the TLM.

It turns out that the number of arbitration checks is in strong correlation with the performance of the model, since these checks typically result in costly context switches in the simulator. Thus, implementing all required arbitration checks is the slowest, but delivers accurate time prediction. The other extreme, the TLM, implements the fewest arbitration checks yielding the highest performance, but results in the worst accuracy.

3.3 Result Oriented Modeling

We will now apply the ROM approach to model the AMBA AHB aiming at 100% accuracy *and* highest simulation performance at the same time.

3.3.1 Assumptions as with TLM

As discussed earlier, ROM is based on separation of computation and communication, and hiding of communication internals from the user. It avoids using signals and individual wires and implements data transfers by use of a single *memcpy* operation. As such, ROM uses the same principles as TLM.

In ROM, the application is only aware of the timing at the boundaries of a user transaction. All activities of the bus model within the user transaction are hidden from the communicating parties. Those are not aware that the transaction is split into multiple bus transactions and cycles, neither that there is arbitration involved.

Only the timing at the boundaries of the user transaction is important for the application. For accurate timing, the start and the end times of each transaction must match the times reported by a bus functional model.

Between the start and end times, ROM can freely rearrange and/or omit internal events and state changes in order to eliminate costly context switches in the simulator.

As in TLM, the main idea for speeding up the simulation is to replace the sequence of wait operations and arbitration checks with one single *wait-for-time* statement. Reducing the number of wait operations is the biggest contributor to increased execution performance. This avoids running the scheduling algorithm in the simulation engine and thus also reduces the number of context switches.

3.3.2 Optimistic modeling

The ROM implements an optimistic approach. When a master requests a user transaction, the earliest finish time for this transfer is calculated and the master waits until that time. The time prediction takes the current state of the bus into account. In case a higher-priority transaction is already active, the wait time is increased for its duration. After the calculated time has passed, the master verifies whether the predicted time is still accurate. If so, the transaction is complete. Note that in this best case scenario only a single wait statement is used (as in the TLM).

With a *disturbing influence* of a higher priority master accessing the bus during a transaction, the predicted time will be too short. Then, ROM recalculates the predicted time and waits for it. This process is repeated until the prediction is verified to be correct.

Note that an optimistic (short) prediction algorithm is necessary to allow for corrections. With a pessimistic (too long) prediction, a correction would need to go back in time, which obviously is not possible.

3.3.3 Preemption

To compare ROM against the layered models, we will analyze the case of a bus preemption in more detail, as shown in Figure 5.

In (a) BFM, a burst transaction starting at t_0 is preempted at t_1. The higher priority transfer completes at t_3 when the preempted transfer resumes, terminating finally at t_4. Both masters perform arbitration checks for every bus cycle, a total of 32 in this example.

In (b) TLM[3], the low priority transaction is not properly preempted and still ends at t_2 (not at t_4). Instead, the high priority transaction is delayed until t_2 and ends at t_4 (not at t_3). Clearly, the abstract TLM is highly inaccurate in the finish times of both transfers, but executes fast. Only two arbitration checks are performed.

In (c) ROM, the inaccuracies of the TLM are corrected by 3 additional arbitration checks. The low priority transfer is initially predicted to finish at t_2. Then, it detects that it has been preempted at t_1 and recalculates its finish time for $t_3 - t_1$ time units later at t_4. The high priority master wakes up at t_3 and terminates its transaction, since it was not preempted. At t_4, the low priority master wakes up, verifies that no other preemption has occurred, and thus completes its transfer.

[3]For brevity, we will omit the similar ATLM case here.

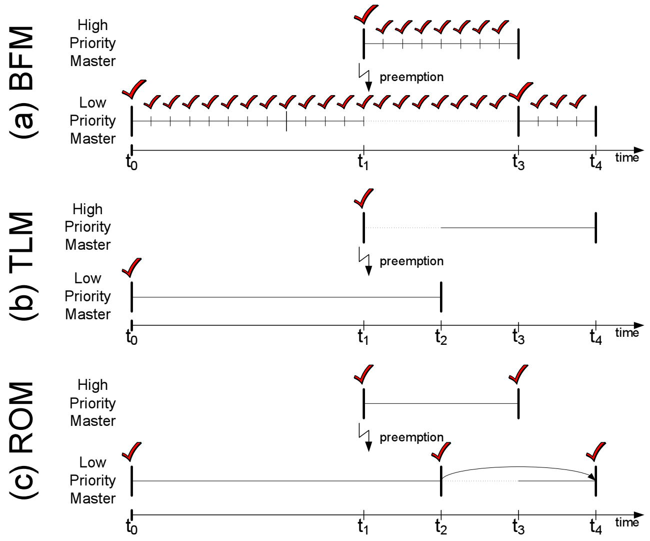

Figure 5: Preemption in BFM, TLM, ROM

Note that the final two arbitration checks performed by the ROM are inexpensive because no further waiting is necessary and no context switch occurs.

	BFM	TLM	ROM
Arbitration Checks	32	2	5
(in percent)	100%	6.3%	15.6%

Table 1: Preemption complexity comparison.

Table 1 compares the arbitration checks performed by the models. The number of checks in ROM is close to the TLM case and an order of magnitude lower than in the BFM.

3.3.4 Multiple prediction updates

In general, multiple prediction updates are necessary if multiple preemptions occur. Figure 6 shows an example where a long transaction is frequently preempted. Although this transfer is preempted 15 times, only 4 prediction updates are needed by the ROM. A closer looks shows 8 preemptions during the initially predicted period, 4 in the next, then 2 and finally only one. This exponential drop indicates that, for most transfers, only very few prediction updates are expected, even under high bus load.

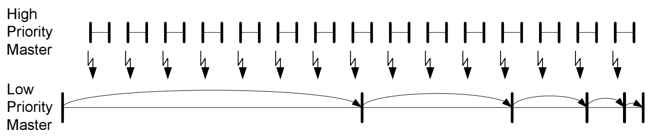

Figure 6: Exponentially decreasing number of prediction updates.

3.3.5 Complexity Considerations

It should be noted that the advantages of ROM come at the price of a more complex model implementation. The BFM and TLM implementations, on one hand, incrementally advance time and can therefore use step-by-step decisions. ROM, on the other hand, implements all bus scheduling decisions explicitly at the boundaries of a user transaction. This requires the model to keep track of outstanding transactions, and reevaluate decisions if they were overly optimistic, requiring a higher effort from the model developer.

4. EXPERIMENTAL RESULTS

In order to validate the benefits of the proposed ROM approach, we have implemented the AMBA AHB using the four models discussed above. We have analyzed the performance of the models in two test setups, a single and a multi master scenario.

4.1 Single Master Setup

To measure the best case for the ROM where no prediction updates are necessary, we have connected a single master to a single slave via the AHB. The master issues user transactions repeatedly, with no delay in between. We have measured the simulation time on a Pentium 4 at 2.8 GHz.

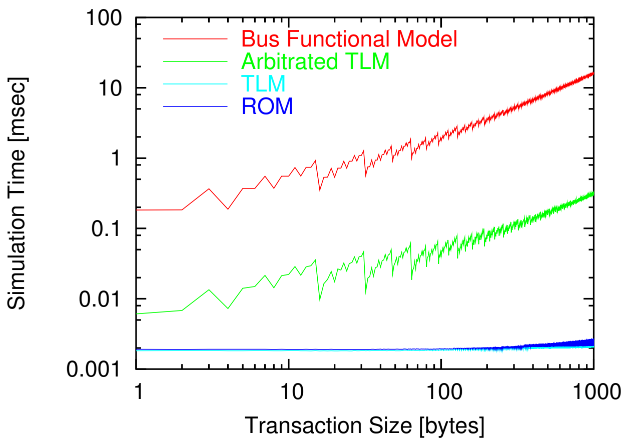

Figure 7: Transfer time for single master setup.

Figure 7 shows the ROM model as fast as the TLM. Both are three orders of magnitude faster than the BFM. This shows the best possible performance for the ROM. Since only a single master is connected to the bus, there is no disturbing influence, and hence no prediction updates are necessary. Note that both, the ROM and the TLM, are independent of the transfer size, since essentially only one wait and a *memcpy* operation are needed.

4.2 Multiple Master Setup

In the second test setup, we have examined the performance with prediction updates using two masters and two slaves. The high priority master puts an equally distributed base load of 33% on the bus by sending 8-beat burst transactions. The low priority master issues transactions of increasing size without a delay in between, as before[4].

[4]For a fair comparison, we also ensure that all models transfer the same amount of user transactions.

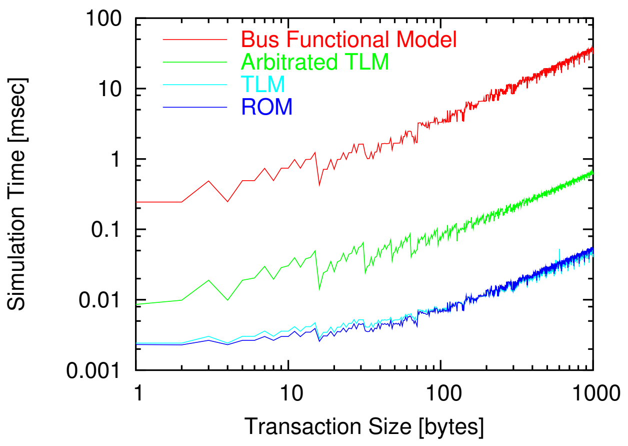

Figure 8: Transfer time with 33% base utilization.

Figure 8 shows the time to simulate the low priority master over an increasing message size while the high priority master is running at the same time. Again, both ROM and TLM are equally fast, three orders of magnitude faster than the BFM, and one order of magnitude faster than the ATLM. However, the simulation time now increases with the transaction size. This is due to the fact that, with an increasing duration of a transaction, also the number of preempting requests per transaction increases. Hence, the amount of work to simulate both the high and the low priority master increases.

4.3 Prediction Updates

It is interesting that even though the ROM has to perform an increasing number of prediction updates with the increased size, the ROM performs as fast as the TLM, even for large transactions. Simulating the preempting transactions of the high priority master (which is done equally for the ROM and TLM) dominates over the additional work of the ROM for the prediction updates.

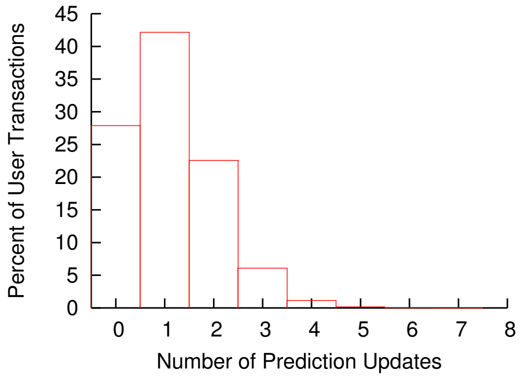

Figure 9: Histogram of prediction updates for random transactions at 50% contention.

As discussed in Section 3.3.4, we expect an exponentially decreasing number of prediction updates for a linear distribution of preemptions through higher masters. In order to verify this very low number of prediction updates, we have established a random transfer setup. Two masters transfer transactions of random size (1-200 bytes) with a random delay in between. 50% bus contention is controlled through the maximum delay between transfers and the maximum size.

Figure 9 shows a histogram of the number of prediction updates per low priority transaction over a set of 100000 transfers. Most transactions require only a single prediction update, despite the high amount of contention. As expected, the number of transactions with more than one prediction update reduces exponentially. Only 1.1% require 4 prediction updates. Note that 27.5% of the transactions complete without a single update. These are mainly small transactions (58% of the transactions with 0 updates are 50 bytes or smaller in size). This distribution clearly shows that the excellent performance of ROM can be generally expected.

4.4 Timing Accuracy

Knowing now that ROM is equally fast as TLM, we need to analyze the accuracy of the models with respect to timing. As discussed above, the start and end times of a transaction are of importance to the designer. The start time is automatically correct for each model, so we have measured the duration of a large number of individual random transfers and have compared those against the cycle-accurate BFM.

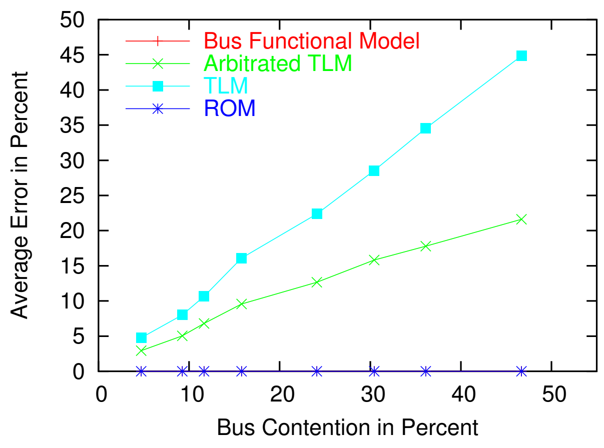

Figure 10: Accuracy comparison for unlocked transfers of the high priority master.

Figure 10 shows the average error in transaction duration for the high priority master over a varying degree of bus contention. As targeted, the ROM shows 0% error for all measurements, lying right on top of the x axis (same as the BFM). In contrast, the TLM and ATLM show significant error rates, linear increasing with growing bus contention. At 45% contention, the TLM reaches 45% error, making any system timing analysis based on TLM questionable. The ROM, however, shows 100% accuracy, enabling fast *and* accurate design space exploration.

5. CONCLUSION

In this paper, we have introduced a novel modeling concept, Result Oriented Modeling (ROM), and its application to the modeling of communication in SoC design. ROM is an abstract modeling approach similar to TLM that hides internal states and minimizes them to gain execution speed. Moreover, ROM is based on an optimistic paradigm. It predicts the end result at the beginning. Under a disturbing influence, corrective measures are taken at the end, in order to adjust the prediction for 100% accuracy.

We have implemented the ROM concept for an AMBA AHB example and have compared the new model against traditional layer-based models. Detailed analysis shows that the cost of corrective measures is low due to an exponential decreasing number of necessary prediction updates.

Our experimental results demonstrate the tremendous benefits of ROM. While the traditional models suffer from a significant speed/accuracy tradeoff, ROM delivers highest speed and 100% accuracy at the same time. This enables true design space exploration at the abstract system level.

For the future, we plan to apply the ROM approach to other bus architectures and applications beyond communication modeling.

6. REFERENCES

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

[2] M. Coppola, S. Curaba, M. Grammatikakis, and G. 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.

[3] D. D. Gajski, J. Zhu, R. Dömer, A. Gerstlauer, and S. Zhao. *SpecC: Specification Language and Design Methodology*. Kluwer Academic Publishers, 2000.

[4] A. Gerstlauer, D. Shin, R. Doemer, and D. Gajski. System-Level Communication Modeling for Network-on-Chip Synthesis. In *Proceedings of ASPDAC*, Shanghai, China, January 2005.

[5] T. Grötker, S. Liao, G. Martin, and S. Swan. *System Design with SystemC*. Kluwer Academic Publishers, 2002.

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

[7] M. Lajolo, C. Passerone, and L. Lavagno. Scalable Techniques for System-level Co-Simulation and Co-Estimation. *IEE Proceedings - Computers and Digital Techniques*, 150(4):227–238, July 2003.

[8] S. Pasricha, N. Dutt, and M. Ben-Romdhane. Fast exploration of bus-based on-chip communication architectures. In *CODES and ISSS*, Stockholm, Sweden, September 2004.

[9] 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.

[10] 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.

[11] R. Siegmund and D. 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.