Quantitative Analysis of the Speed/Accuracy Trade-off in Transaction Level Modeling

Gunar Schirner and Rainer Dömer
Center for Embedded Computer Systems
University of California, Irvine

The increasing complexity of embedded systems requires modeling at higher levels of abstraction. Transaction Level Modeling (TLM) has been proposed to abstract communication for high-speed system simulation and rapid design space exploration. Although being widely accepted for its high performance and efficiency, TLM often exhibits a significant loss in model accuracy.

In this article, we systematically analyze and quantify the speed/accuracy trade-off in TLM. To this end, we provide a classification of TLM abstraction levels based on model granularity and define appropriate metrics and test setups to quantitatively measure and compare the performance and accuracy of such models.

Addressing several classes of embedded communication protocols, we apply our analysis to three common bus architectures, the industry-standard AMBA Advanced High-performance Bus (AHB) as an on-chip parallel bus, the Controller Area Network (CAN) as an off-chip serial bus, and the Motorola ColdFire Master Bus as an example for a custom embedded processor bus.

Based on the analysis of these individual busses, we then generalize our results for a broader conclusion. The general TLM trade-off offers gains of up to four orders of magnitude in simulation speed, generally however, at the price of low accuracy. We conclude further that model granularity is the key to efficient TLM abstraction, and we identify conditions for accuracy of abstract models. As a result, this article provides general guidelines that allow the system designer to navigate the TLM trade-off effectively and choose the most suitable model for the given application with fast and accurate results.

Categories and Subject Descriptors: I.6.4 [**Simulation and Modeling**]: Model Validation and Analysis

General Terms: Design

Additional Key Words and Phrases: System Level Design, Transaction Level Modeling, System-on-Chip

1. INTRODUCTION

Designing embedded systems and System-On-Chip (SoC) becomes increasingly challenging. The design space to be explored grows with increasing complexity, while at the same time shorter product life cycles require a shorter time-to-market. Addressing this gap has been the aim of system-level research. As one main ap-

Author's addresses: Gunar Schirner, Center for Embedded Computer Systems, UC Irvine, Irvine, CA 92697-2625, USA; email: hschirne@uci.edu; Rainer Dömer, Center for Embedded Computer Systems, UC Irvine, Irvine, CA 92697-2625, USA; email: doemer@uci.edu

proach, abstract models have been introduced to tackle the design complexity. For one, abstract models exhibit tremendous gains in simulation speed, allowing fast validation and extensive design space exploration.

For communication in particular, Transaction Level Modeling (TLM) has been proposed [Grötker et al. 2002]. TLM abstracts the communication in a system to whole transactions, abstracting away low level details about pins, wires and waveforms [Cai and Gajski 2003][1]. This results in models that execute dramatically faster than synthesizable, bit-accurate models. This benefit, however, usually comes at the price of low accuracy.

1.1 TLM Trade-off

In general, TLMs pose a trade-off between an improvement in simulation speed and a loss in accuracy, as illustrated in Figure 1. The trade-off essentially allows models at different degrees of accuracy and speed. However, having both high speed and high accuracy at the same time is typically not possible. High simulation speed is traded in for low accuracy, and a high degree of accuracy comes at the price of low speed. Models with this trade-off fall into the gray area of the diagram. Models in the dark area are obviously existent, but practically unusable, whereas models in the white area are highly desirable but typically not achievable. Although TLM has been generally accepted as one solution to tackle SoC design complexity, the TLM trade-off however, has not been examined in detail.

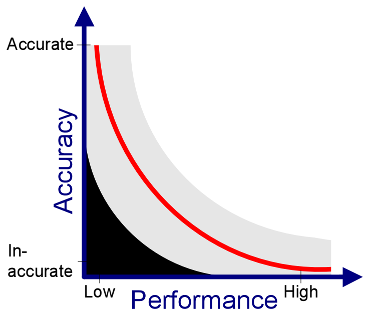


Fig. 1. Transaction Level Modeling Trade-Off.

1.2 Problem Statement

For this work, we define the problem as follows: a quantitative analysis of the trade-off between simulation performance and accuracy is needed. Definitions of generic metrics and a test framework, applicable to a range of communication protocols, are required to perform the analysis. The immediate analysis goal is to confirm the existence of the TLM trade-off. As a broader goal, we also expect conclusions on (a) classification of abstraction levels, (b) guidelines for model designers in efficiently abstracting communication, and (c) guidance of communication model users in the selection of a suitable model for a given design task.

1.3 Overview

In this article, we systematically study and analyze the TLM trade-off quantitatively. More specifically, we quantify the performance gains of TLM and measure the loss in accuracy for a wide range of bus systems. For our analysis, we define in Section 4 proper metrics and test setups for measuring the performance improvement and the accuracy loss. For each bus system, we use models at different

[1]General modeling of SoCs consists of two parts, computation and communication. In this article, we focus only on modeling of the communication.

abstraction levels as defined in Section 3. In particular, we use two classes of TLMs (ATLM and TLM), and compare them against a fully accurate Bus Functional Model (BFM) as a reference.

Our measurements are based on examples from three different bus categories. In Section 5, we analyze the Advanced High-performance Bus (AHB) of AMBA [ARM 1999], as a representative of parallel on-chip bus systems with centralized arbitration and multiplexed interconnection scheme. AMBA is a widely used and industry-accepted standard for on-chip bus systems. For the second category of off-chip serial busses with distributed arbitration, we investigate the Controller Area Network (CAN) [Bosch 1991] in Section 6. This bus system dominates in automotive applications. Third, we analyze in Section 7 the category of custom processor-specific busses that are typically much simpler than the general purpose standard busses. Here, we have chosen the Motorola ColdFire Master Bus [Motorola 1997] that is used by the popular ColdFire MCF5206 processor.

The analysis of the first two busses is based on our previous work with the AMBA in [Schirner and Dömer 2006] and CAN in [Schirner and Dömer 2005a]. Whereas these previous publications focused on each bus individually, we will in this article combine and generalize the results for different categories and include the ColdFire Master Bus as an example of a third category. Most importantly, based on the results of this range of examples, we will then generalize our TLM analysis in Section 8 and derive general conclusions in Section 9.

2. RELATED WORK

System level modeling has become an important research area that aims to improve the SoC design process and its productivity. Languages for capturing SoC models have been developed, examples are SystemC [Grötker et al. 2002] and SpecC [Gajski et al. 2000]. Using TLM [Grötker et al. 2002] for capturing and designing communication architectures has received much attention. Cai and Gajski [2003] provide an initial taxonomy of TLM. [Rose et al. 2005] define a standard for transaction level modeling in SystemC.

Sgroi et al. [2001] address the SoC communication with a Network-on-Chip approach. Here, communication is partitioned into layers following the OSI structure. Software reuse is promoted with an increase of abstraction from the underlying communication. Siegmund and Müller [2001] 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. Brem and Müller [2003] describe how the CAN bus is modeled using SystemC^{SV}. The work also shows the three abstraction levels, but does not give any experimental results on performance or accuracy.

In [Caldari et al. 2003] Caldari et al. describe the results of capturing the AMBA rev. 2.0 bus standard in SystemC. The bus system has been modeled at two levels of abstraction, first a bus functional model on RTL level, and second a model on TLM level. Their TLM reached a speedup of 100 over the RTL model. Coppola et al. [2003] also propose abstract communication modeling. They present the IPSIM framework and show its efficient simulation. Gerstlauer et al. [2005] describe a layered approach and propose models that implement an increasing number of ISO OSI layers [ISO 1994]. Simulation speedup up to 100x is shown, but the accuracy analysis is limited.

Haverinen et al. [2002] describe in a white paper three TLMs with increasing abstraction for the OCP-IP protocol. Only their most detailed TL-1 is cycle accurate. They do not show an accuracy analysis for the more abstract models. Pasricha et al. [2004] describe an 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 reported[2].

3. TRANSACTION LEVEL MODELING

TLM allows a wide variety of modeling styles and abstractions. It is not clearly defined in the literature, as also stated by [Cai and Gajski 2003] in their approach of structuring the models.

For our modeling, we focus on the *granularity* of data and arbitration handling. We define three classes of granularity applicable to any bus protocol, and match these granularity classes to three model types. Figure 2 shows the granularity levels with respect to time and indicates the correlation to models and layers.

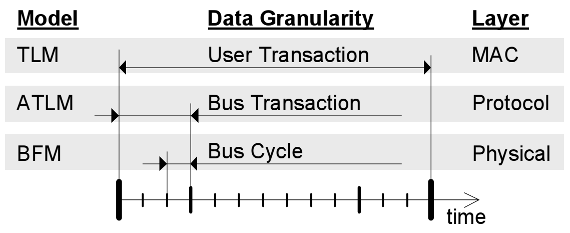


Fig. 2. Model classes and their granularity.

A *user transaction* is the most coarse grain element of transferring a contiguous block of bytes with arbitrary length. It is split into *bus transactions*, which are bus transfer primitives, such as a word transfer. Each bus transaction is usually processed in several *bus cycles*, which represent the finest granularity in our modeling.

Our classes of granularity follow the layers defined in the ISO OSI reference model [ISO 1994]: the media access control (MAC), the protocol sublayer, and the physical layer. Each considered layer handles data and arbitration at its own granularity.

The *media access layer* provides a transmission service for a user transaction, which is a contiguous block of bytes. 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 in turn uses the *physical layer*, which provides services to sample and drive individual bus wires at bus cycle granularity.

[2]Our results show a speedup of up to four orders of magnitude.

For dealing with the description complexity, we chose a layered architecture for our bus models similar to [Gerstlauer et al. 2005]. Using a system level modeling approach, each layer can be implemented as a separate channel using a system level design language (SLDL)[3]. Then, different models can be easily created by composing the channels hierarchically.

According to our classes of granularity, we consider models at three different abstraction levels; the *Transaction Level Model* (TLM) that models user transactions, the *Arbitrated Transaction Level Model* (ATLM) at bus transaction granularity, and the bus cycle accurate *Bus Functional Model* (BFM).

3.1 Transaction Level Model (TLM)

The TLM[4] 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 *wait-for-time* statement, covering the whole user transaction. Arbitration is not modeled. Instead, concurrent access is resolved using a semaphore once per user transaction. The semaphore-based contention resolution depends on the simulation environment and is independent of the arbitration of the actual bus protocol.

3.2 Arbitrated Transaction Level Model (ATLM)

The ATLM simulates the bus access with bus transaction granularity (e.g. AHB bus primitives). It uses the MAC layer to split user transactions into bus transactions and implements an own abstracted protocol layer. The ATLM accurately models the bus arbitration for each bus transaction. We implement the arbitration and the bus without an own flow of execution to maximize simulation performance. Although this model correctly models arbitration, it is not pin-accurate and not cycle-accurate in all cases.

3.3 Bus Functional Model (BFM)

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 has the capability to take arbitration decisions on a bus cycle granularity. This model may include additional hardware, such as an arbiter, to correctly implement the bus standard.

3.4 Comparison with other TLM Abstractions

In order to relate our TLM abstraction levels to other abstraction schemes, we will briefly compare our models to the OCP-IP and and SystemC TLM standards.

In comparison to the OCP-IP TLM definitions [Haverinen et al. 2002], our models range between TL0 and TL3. Our BFM correlates most closely to TL0, since it models all bus wires, implements active bus components if applicable (e.g. mul-

[3]We have used SpecC [Gajski et al. 2000] as the SLDL of choice, we could have used SystemC just as well.

[4]In the general sense we consider TLM to be a class of models and our bus models are a part of this class. To be consistent with our previous publications however, we have kept TLM also as a name of our most abstract model.

tiplexers of AHB) and is based on an explicit clock. Our ATLM compares to TL2, since it uses bus transactions and is cycle-approximate in the general case[5]. Our TLM bus model lies between TL2 and TL3. It is more abstract than TL2, since it is based on user transactions (messages in the OCP-IP). On the other hand, it provides cycle-approximate timing, hence it is more detailed than TL3.

A comparison with the SystemC TLM standard [Rose et al. 2005] is more difficult, because the version 1.0 does not define the SystemC abstraction levels in detail. Our BFM matches the Cycle Callable (CC) model, since it is cycle-accurate and bit-accurate (CABA). In addition to the CC properties, our BFM is pin-accurate (PA) as well. Both our ATLM and the TLM could be called a Programmers View + Time (PVT) model, since both provide cycle-approximate (CX) timing.

4. METRICS AND MEASUREMENT SETUP

Given the model classification, we will now describe our metrics and setup for analyzing our models. We focus on two aspects. First, we look at the simulation *performance*, since a performance gain is the main premise of TLM. Second, we evaluate the *timing accuracy*.

4.1 Performance

Our metric for the performance is the simulation bandwidth. The bandwidth is the amount of data transferred over the simulated bus in one second of real-time.

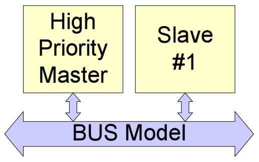


Fig. 3. Single master setup for performance measurements.

We have measured the simulation performance of each model in a minimal scenario with one master and one slave (Figure 3). A transaction with random content is performed repeatedly, without any delay in between. We have measured the simulation time (also referred to as real time or wall clock time) over a set of repeated iterations (e.g. 5000) and computed the simulation bandwidth. In order to estimate the scalability, we repeat this test for an increasing size of the user transaction.

Besides the data transfer and the controlling loop, the test masters and slaves do not execute any other code. This minimizes their impact on the overall simulation time and allows us to examine solely the performance of the bus models. All tests have been executed using the SpecC V 2.2.0 discrete event simulator using Quick Threads on a Pentium 4 (2.8 GHz) PC running Red Hat Enterprise WS 3.

4.2 Accuracy

Accuracy can be segregated into many aspects. For the purpose of this article, we will distinguish three aspects of accuracy: functionality, represented feature detail, and timing[6]. For our analysis, we will fix the first, vary the second, and measure the third aspect.

[5]The ATLM can be cycle-accurate for protocols without preemption, as we will show later.
[6]We omit one dimension of model accuracy, the "coding quality", due to the difficulty of a realistic measurement. In order to keep this factor as constant as possible, all models have been implemented with the same optimization effort by the first author.

The first aspect of functional accuracy requires that the data transmitted by the sender reaches correctly the receiver. Since this is a necessary requirement for a functional simulation, all our models are functionally accurate. The second aspect is the represented detail level, which states how many features of the bus are actually present in the model. We vary this aspect in our model under test, ranging from accurately modeled pins and waveforms in the BFM up to a single *memcpy* and *wait-for-time* statement in the TLM[7]. The third aspect is the accuracy in timing which we measure in our tests. To compare the models, we use the timing of user transactions.

4.2.1 *Timing Accuracy Metrics.* The relevant measurements for expressing the accuracy of a model depend on the prediction goal of the simulation. We have identified two goals: first, the prediction of the application *latency* due to an individual bus access, and second, the prediction of the application *finish time*. For these goals, we use two metrics. First, by analyzing the *individual* duration of a user transaction, the application latency due to a bus access can be inferred. Second, the *cumulative* duration, which is the sum of all user transactions, shows how much the application finish time is delayed by the performed communication. For our tests, we prefer the cumulative transfer time over the actual finish time, since the latter includes the constant computation time between transactions (simulated by a delay), which is independent of the utilized bus model.

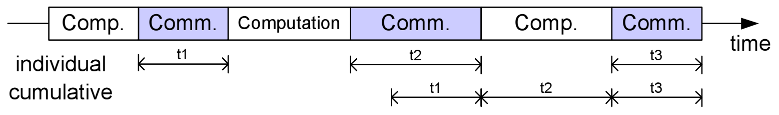


Fig. 4. Cumulative and individual transfer time.

Figure 4 shows an example of a bus node's activity over time which progresses along the x-axis. Any process alternatively performs computation or communication, the latter of which we are interested in. For the individual duration, we analyze the transfers t_1, t_2 and t_3 separately. For the cumulative analysis, we use the sum of the three transfers.

For the purpose of analyzing the accuracy based on the individual and cumulative duration, we define for this article the *duration error* as the percentage error over the bus standard[8]:

$$\begin{aligned} d_{std} :& \quad \text{duration as per standard} \\ d_{test} :& \quad \text{duration in model under test} \\ error_i =& \ 100 * \frac{|d_{test} - d_{std}|}{d_{std}} \qquad (1) \end{aligned}$$

Given this error definition, a timing accurate model exhibits 0% error. Note also that a particular model may have an error of more than 100% (i.e. the model under test shows more than twice the simulated time).

[7]A detailed table of represented features is given later in Section 6 for the CAN bus.

[8]For practical purposes, we use a two step approach. We first implement and validate the BFM according to the bus standard. Then, we use the BFM to obtain the reference timing.

4.2.2 *Bus Contention Metrics.* Our models differ in the granularity of data and arbitration handling. We therefore expect a significant correlation between bus contention and accuracy. Thus, we use contention as an input to our measurements.

For our tests, we examine the status of a user transaction to determine the bus contention. This makes the contention definition independent of the actual bus implementation and can be applied to different bus systems. For this paper, we define the bus contention as the percentage overlap between user transactions:

$$contention = 100 * \frac{\text{bus cycles with two or more active user transactions}}{\text{bus cycles with at least one active user transaction}} \tag{2}$$

Figure 5 depicts example contentions of 0%, 25% and 50%. The low priority transaction starts delayed due to an ongoing high priority transaction, which is considered bus contention in our definition.

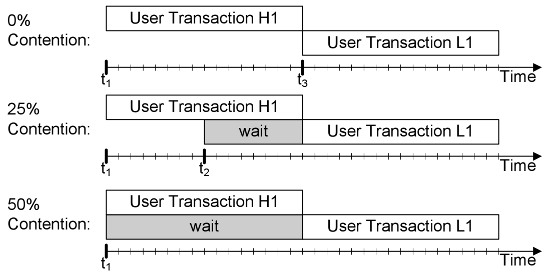


Fig. 5. Bus contention.

4.2.3 *Accuracy Measurement Setup.* The actual timing accuracy highly depends on the test setup and the communication patterns of the involved applications. Therefore, we define a generic test setup and a procedure that covers a range of applications, so that the designer can derive the expected accuracy for her/his particular setup.

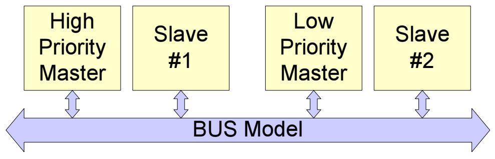


Fig. 6. Dual master setup for accuracy measurements.

We use a generic test setup with two masters and two slaves connected to the same bus, as shown in Figure 6. Each master transfers a predefined set of 5000 user transactions to a slave target. Each transaction varies linear randomly in the base address, size (1 to 100 bytes), content[9], and the delay to the next transaction. This delay simulates the application's computation. It ranges from a zero delay to a maximum delay value that depends on the desired contention (see discussion below). The list of all transactions is created before the actual test by using a random number generator with a fixed start seed.

[9]Please note that for some protocols, e.g. the CAN protocol, the content of the transaction can influence the transaction duration.

During the simulation, the start time and the duration is recorded for each individual user transaction and each master. The test is repeated for each implemented bus model. Since each model transfers the same set of user transactions, the results are comparable and can be analyzed together.

We have repeated the described test for different levels of bus contention. Since the bus contention cannot be controlled directly, we have varied instead the maximum delay between user transactions for each test run. This results in a varying bus utilization per master. Since the two masters share the same bus in our setup, the utilization correlates to the amount of bus contention. We have measured the resulting amount of bus contention according to our previous definition using the BFM, by checking for each clock cycle whether one or two user transactions are active. After we have determined the maximum delay that produces a desired contention, we use the created transaction set for measuring the abstract models.

5. AMBA

The first bus system is an example for an on-chip bus system with a centralized arbitration scheme [Schirner and Dömer 2006]. We use the Advanced Microprocessor Bus Architecture (AMBA) for our analysis.

5.1 Introduction

AMBA defined by ARM [ARM 1999] is a widely used and industry accepted standard for an on-chip bus system. The AMBA standard contains a group of busses, which are used hierarchically, as shown in Figure 7. We focus on the Advanced High-performance Bus (AHB), a system bus designed for connecting high-speed components including ARM processors.

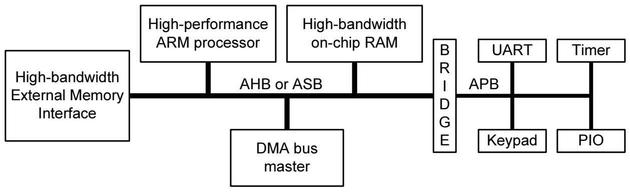
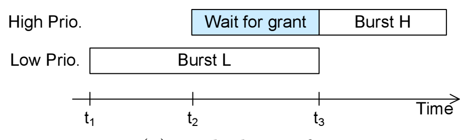

High-performance ARM processor
High-bandwidth on-chip RAM
High-bandwidth External Memory Interface
AHB or ASB
DMA bus master
BRIDGE
UART
Timer
APB
Keypad
PIO

Fig. 7. AMBA bus architecture (Source [ARM 1999]).

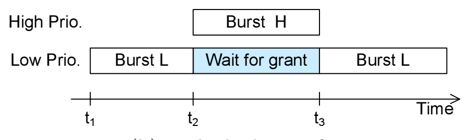
The AHB is a multi-master bus that operates on a single clock edge and employs a centralized arbitration scheme. High performance is achieved by a pipelined operation that overlaps address and data phases, as well as by the usage of burst transfers. Split and retry transfers allow the slave to free the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

5.1.1 *Locked versus Unlocked Transfers.* The AHB allows two operation modes, *locked* and *unlocked* transfers. These differ in whether preemption of a burst transfer is allowed. An example is shown in Figure 8.

In the locked mode in Figure 8(a), a ongoing burst cannot be preempted by a higher priority bus transaction. The high priority transaction released at t_2



(a) Locked transfer



(b) Unlocked transfer

Fig. 8. AMBA AHB operation modes.

is delayed until the current low priority bus transaction finishes at t_3. In the unlocked mode in Figure 8(b), the high priority transaction preempts the ongoing low priority bus transaction and gains bus access already at t_2. Since the operation mode influences the granularity of the arbitration decision, we will analyze these two operation modes separately.

5.2 Models

For modeling the AMBA AHB, we have applied the granularity-based approach described in Section 3.

Transaction Level Model (TLM). The TLM is the most abstract model that operates on the granularity of user transactions. It resolves contention independent of the priority arbitration using a semaphore once per user transaction, as described in Section 3.1. It provides basic support for bursts.

Arbitrated Transaction Level Model (ATLM). The ATLM operates on the bus transaction granularity, such as a StoreWord, StoreByte, LoadBurst. It adds support for priority-based arbitration and reflects the effects of pipelined operation. We have implemented two variants of the ATLM that differ in the time frame to collect arbitration requests. On an idle bus, the ATLM (a) collects requests for one clock cycle before making a decision, as required by the standard. The ATLM (b), on the other hand, makes the decision immediately after receiving the first request. Both variants behave identical when the bus is busy: requests are collected while a bus transaction is active, and the highest priority master continues after that.

Bus Functional Model (BFM). The BFM is the bus-cycle accurate and pin-accurate bus model. In order to correctly model the bus architecture, we have implemented several active components, such as multiplexers, an arbiter and an address decoder. On top of the features offered by the ATLM, the BFM supports unlocked (preemptable) transfers. At this point, our model does not support split and retry operations.

We have functionally validated all described models. We have also validated the cycle count timing of the BFM against the standard for all bus primitives, and have compared the waveforms with examples in [ARM 1999; 2003]. Finally, we have ensured that all abstract models show the correct timing in the single master setup.

5.3 Performance Analysis

Our performance measurements (Figure 9) confirm the TLM expectations: the simulation speed increases significantly with abstraction. The performance raises with each TLM abstraction by two orders of magnitude. However, no significant

	BFM	ATLM (a)	ATLM (b)	TLM
Simulation Time [ms]	16.75	0.2137	0.206	0.00246
Sim. Bandwidth [MByte/s]	0.03	2.29	2.37	198
Rel. Speedup over BFM	1	78	81	6802

Table I. Performance comparison for transferring 512 bytes using AMBA AHB models.

performance difference exists between the variants within the ATLMs. The additional abstraction of the (b) variant does not yield a significant speed improvement. Table I compares the performance in detail for transferring 512 bytes.

The TLM executes the fastest among the analyzed models. Its simulation bandwidth increases linearly with the transaction size, since a constant number of operations is executed for each transfer (one *memcpy* and one *wait-for-time*). The ATLMs are two orders of magnitude slower due to the finer granularity of modeling individual bus transactions. Starting with the ATLMs, the graphs exhibit a saw tooth shape due to the non-linear split of user transactions into bus transactions (e.g. 3 bytes are transferred in 2 bus transactions: byte + short, whereas 4 bytes are transferred in 1 bus transaction: a word). The BFM is again two orders of magnitude slower than the ATLMs due to the fine grain modeling of individual wires and additional active components (e.g. multiplexers).

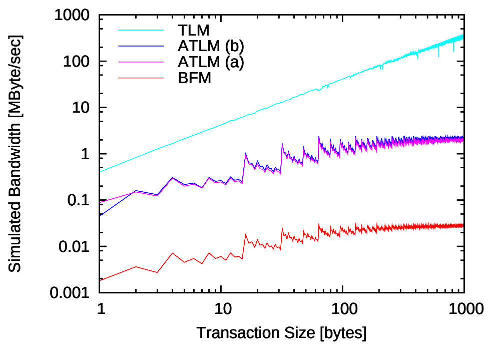


Fig. 9. Performance for the AMBA AHB models.

5.4 Accuracy Analysis

The performance analysis for the AHB has shown the impressive speed benefits of TLM. Now we will evaluate the accuracy reduction, that the designer has to accept for achieving the higher simulation speed. As described before, we analyze *locked* and *unlocked transfers* separately.

5.4.1 *Analysis for Locked Transfers.* With locked transfers, a burst cannot be preempted by a higher priority master. Figure 10 shows the average timing error over a range of bus contention for the high priority master. The x-axis denotes the amount of bus contention in percent, each measurement point for a model reflects the average error over the analyzed 5000 user transactions.

The BFM and the ATLM (a) perform accurately over the whole range of bus contention (their graphs lie on top of the x-axis). Since the test setup uses only locked transfers, a burst is not preemptable and therefore no arbitration test is needed within a bus transaction. The features abstracted away in the ATLM (a) are not exercised. The ATLM (b), which makes an immediate decision and does not collect further requests, shows an inaccuracy of up to 18%. It may mispredict bus access when the two masters attempt a bus access within the same clock cycle.

On the other hand, the ATLM (b) is accurate, when the additional bus request arrives during an active bus transaction. Therefore, the error rate plateaus for bus contentions higher than 35%.

The TLM, which handles contention resolution on user transaction level with a semaphore, performs the least accurate due to the coarse grained decision independent of the master's priority. Its inaccuracy amounts up to 35%.

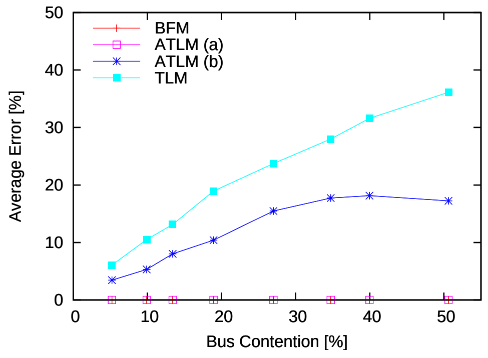


Fig. 10. Individual timing accuracy of locked transfers for the AMBA AHB models.

The measurements show very similar results for the low priority master. Hence, its graph is omitted for brevity, but can be found in [Schirner and Dömer 2005b].

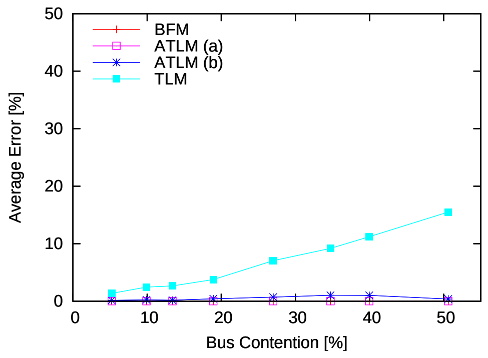


Fig. 11. Cumulative timing accuracy of locked transfers for the AMBA AHB.

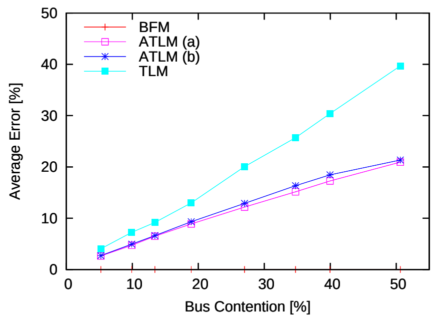
To predict the delay of an application due to bus communication, we have also analyzed the same experimental data in terms of cumulative transfer time. Figure 11 shows the error in the cumulative duration for the high priority master using locked transfers[10].

As in the previous graph, the lines for both BFM and ATLM (a) lie on top of the x-axis. The graph reveals that the mispredictions made by the less accurate ATLM (b) do average out. Both ATLM variants are good predictions for the application finish time. Only the mispredictions of the TLM do not average out, its error increases linearly to 15% for 50% bus contention.

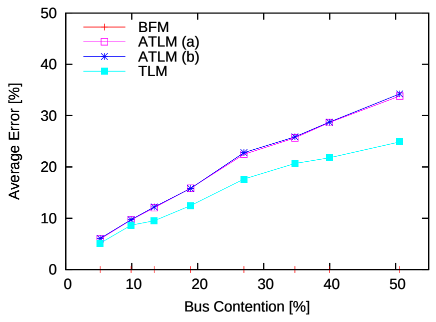
5.4.2 *Analysis for Unlocked Transfers.* We have repeated the same experiment for unlocked transfers. A burst may be preempted in this operation mode by a higher priority bus master, to be resumed later. As before, we have analyzed the accuracy for both: the individual transfer duration and the cumulative transfer time. Both analysis aspects yield similar results, therefore only one - the accuracy based on cumulative transfer time - is shown in Figure 12. Since the results differ by priority, the graphs for the high priority master and the low priority master are shown side by side.

Figure 12 shows that only the BFM yields accurate results. With unlocked transfers, an arbitration decision is also necessary within a bus transaction. Therefore

[10]The graph for the low priority master is omitted, due to its similarity.



(a) high priority master



(b) low priority master

Fig. 12. Cumulative timing accuracy for unlocked transfers for the AMBA AHB.

the ATLM (a) does now show an error of up to 35% for the low priority master. It handles arbitration only at the bus transaction granularity. The difference between the variants of the ATLM becomes insignificant. The ATLMs perform similar for the high and the low priority master.

The TLM can only give a very rough timing estimate. It exhibits a linear increasing error of up to 45% for the high priority master; the test requires arbitration handling for each bus cycle. The error is less for the lower priority master.

5.4.3 *Error Distribution.* Up to now our analysis has focused on the average of errors, which is applicable when predicting the overall system performance. Additionally, the error distribution may be of interest, to judge the range of errors to be expected during simulation. Figure 13 shows a histogram of the transaction durations normalized to the duration in the BFM. A value of 1 indicates an accurate prediction, with a value of 2 the abstract model simulates a transaction with twice the time of the BFM.

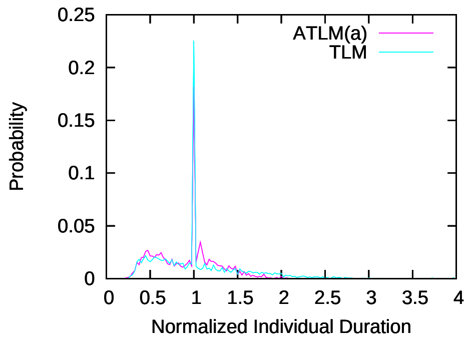


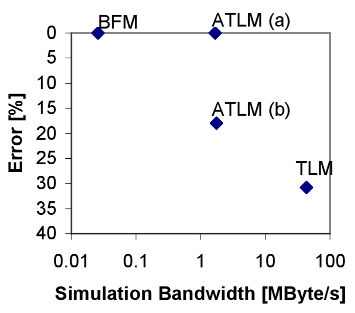
Fig. 13. Histogram of normalized transaction duration.

For simplicity, we only compare the ATLM (a) and the TLM. Both are timing accurate for about 20% of the user transactions. The normalized duration of the ATLM range from 0.25 to 2, the results of the TLM are more wide spread, ranging from 0.25 up to 4. Thus, the ATLM has a tighter error bound.

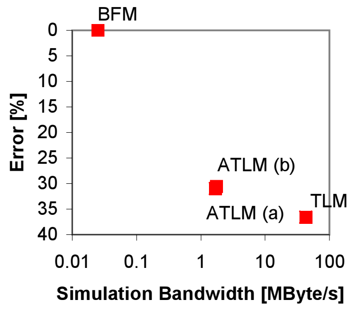
5.5 Summary for the AMBA AHB

Figure 14 summarizes our analysis for the AMBA AHB and depicts the actual TLM trade-off. It shows the detailed relation between performance and accuracy. The x-axis denotes the performance in simulation bandwidth for transferring a 100 byte user transaction. The y-axis denotes the accuracy as the average error in individual timing at 40% bus contention for the low priority master.

Comparing to Figure 1, the Figures 14 show the expected trade-off curve, with



(a) Locked transfers



(b) Unlocked transfers

Fig. 14. AMBA AHB TLM trade-off.

the exception of the ATLM (a) in Figure 14(a), which shows medium bandwidth at 0% error due to the locked transfer mode. Note that in Figure 14(b) the ATLM (a) does not exhibit this exception, lying directly over the ATLM (b).

In summary, our measurements of the AMBA AHB confirm the expectation of a significant speedup with abstraction. The ATLMs are two orders of magnitude faster than the BFM and simulate with more than 1 MByte/s. Decreasing the granularity further to user transactions, as done by the TLM, increases the speed for another two orders of magnitude.

Both variants of the ATLM show the same performance. This indicates that staying within the same granularity level, in this case bus transactions, usually leads to similar performance.

The increasing abstraction, however, also leads to a decreased timing accuracy. The BFM is the slowest model, but always delivers accurate results. The medium speed ATLM model, is accurate for the locked transfers, but is inaccurate for the unlocked transfers (30% error). The fastest model, the TLM, also is most inaccurate (30% error for locked and 37% for unlocked transfers).

The ATLM (b) variant, which performs a reduced arbitration, is not efficient. Although it is not faster than the ATLM(a) it does produce inaccurate results even for the locked transfers (17% error).

Based on our performance and accuracy analysis, Table II lists the fastest model that yields acceptable results for a given situation and simulation focus.

Environment Condition	**Model**	**Speedup**
• single master bus • no bus contention	TLM	10^4
• only locked transfers • unlocked transfers, low contention	ATLM	10^2
• unlocked transfers, high contention	BFM	10^0

Table II. AMBA AHB model selection

All our AHB models are accurate in case of zero bus contention. Therefore, the TLM should be chosen in an architecture with a single master, or when no bus contention is expected. Then, it delivers accurate results the fastest. A system that only uses locked transfers can be accurately simulated by the ATLM. Its results are also acceptable for unlocked transfers in systems with a low bus contention. However, when simulating unlocked transfers under high bus contention only the BFM yields accurate results.

We have modeled the AMBA AHB as a representative for the category of on-chip parallel busses with centralized arbitration. We expect that our results are a good indicator for the behavior of models of other busses in this category, e.g. the IBM CoreConnect Processor Local Bus [IBM 2004].

6. CAN

Our second bus example [Schirner and Dömer 2005a] falls into the category of an off-chip serial bus with decentralized arbitration. The Controller Area Network (CAN) is a serial communication protocol introduced by the Robert Bosch GmbH [Bosch 1991], that was designed for automotive applications.

6.1 Introduction

CAN is a serial multi-master broadcast bus. Frames, with up to 8 bytes user data, are received by all bus nodes and distinguished by the frame identifier. Each bus node decides using local rules whether or not to process a frame. The frame identifier also serves as a priority. If multiple senders simultaneously attempt a transmission, the collision free CSMA/CA arbitration will guarantee that the highest priority frame will succeed undisturbed.

After transmitting the start of frame bit (see Figure 15), the frame identifier is transmitted with the most significant bit first as a sequence of recessive (1) and dominant (0) bus states. During transmission, each sender compares the sent and received signal. If a sender has sent a recessive bit but detects a dominant bit, it will back off from transmission because another sender must have started a higher priority frame.

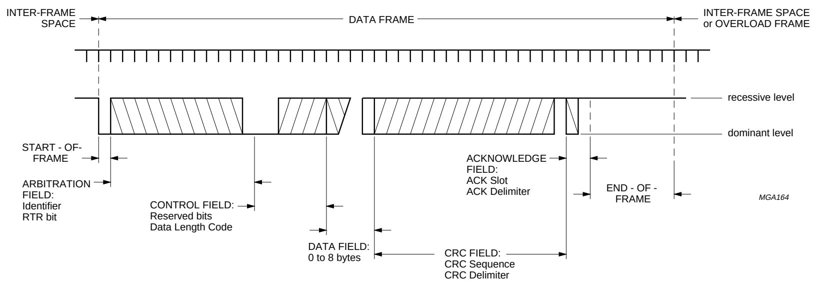

INTER-FRAME SPACE
DATA FRAME
INTER-FRAME SPACE or OVERLOAD FRAME
recessive level
dominant level
START - OF- FRAME
ARBITRATION FIELD: Identifier RTR bit
CONTROL FIELD: Reserved bits Data Length Code
DATA FIELD: 0 to 8 bytes
CRC FIELD: CRC Sequence CRC Delimiter
ACKNOWLEDGE FIELD: ACK Slot ACK Delimiter
END - OF - FRAME
MGA164

Fig. 15. CAN Data Frame (Source [Philips]) .

In order to allow detection of corrupted data, each CAN frame includes a 15-bit CRC. In case of a CRC mismatch, a retransmission of the frame is triggered. The protocol also defines elaborate error detection and error confinement rules for protection against faulty bus nodes.

The CAN serial protocol operates without a centralized clock. Each bus node synchronizes on the bit stream of the sender. A bit stuffing rule guarantees sufficient edges for this synchronization. After transmitting 5 bits of equal polarity, a bit of opposite polarity is introduced.

For modeling a CAN bus, the following features are candidates for 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

6.2 Models

We have applied our granularity-based abstraction to model the CAN. For each model, we have selected a subset of the above listed features.

Transaction Level Model. The most abstract model is identical to the TLM of the AHB. Contention resolution is implemented on the user transaction granularity by using a semaphore (which ignores the frame identifier).

Arbitrated Transaction Level Model. The ATLM simulates arbitration accurately for each bus transaction (CAN frame) based on the frame identifier. It collects all requests during the start of frame, and proceeds with the highest priority frame.

Again, we have defined two variants for the ATLM model. The ATLM (a) performs a bitwise inspection of the frame in order to calculate the CRC and handle stuff bits: a stuff bit is inserted/removed each time 5 bits of equal polarity are found. Note that due to the bit stuffing, the physical frame length depends on the frame content.

The ATLM (b) does neither calculate the CRC nor does it handle stuff bits. It avoids the costly bit inspection and is expected to execute faster than the ATLM (a), however at the cost of accuracy.

Bus Functional Model. The BFM implements all features of the specification. It protects the data by the CRC, handles stuff bits and performs arbitration. The frame data is sent and received serially and the nodes clock is synchronized to the bit stream according to [Bosch 1991] and [Hartwich and Bassemir 1999].

Table III summarizes the features included in each model.

Feature	**BFM**	**ATLM (a)**	**ATLM (b)**	**TLM**
Serial transmission, bit sync	yes	no	no	no
Error detection, confinement	yes	no	no	no
CRC calculation, bit stuffing	yes	yes	no	no
CSMA/CA arbitration	yes	yes	yes	no

Table III. Summary of features supported or abstracted away in the CAN models.

6.3 Performance Analysis

As with the AMBA AHB, we analyze the CAN models in terms of performance and accuracy. The results of our performance measurements in terms of simulation bandwidth over an increasing user transaction size are shown in Figure 16. In addition, Table IV compares the performance of the models in detail for a user transaction of 16 bytes.

Our performance measurements of the CAN show a similar dramatic increase in simulation speed as seen for the AHB. The TLM shows the highest simulation bandwidth, which increases linearly due to the constant number of operations. It achieves 12 MBytes/s when using 16 byte user transactions.

The ATLM (b) is the next slower model (12x). It does not model bit stuffing and CRC. Since the ATLM models individual bus transactions, a step is noticeable in the graph for each 8 bytes - an additional CAN message is needed for transferring the user data. The execution time increases linearly with the amount of bus transactions, reaching a bandwidth of 1 MByte/s.

The ATLM (a) performs 8 times slower than the ATLM (b), since it inspects every bit of the message for the bit stuffing and the CRC calculation.

The BFM is another two orders of magnitude slower than the ATLM (a), due to the additional computational effort for the fine grained serial transmission and the bit synchronization. In addition, the structure of the implementation reduces the performance. For each bus node two extra threads of execution are required, one for the bit stream processor and one for the bit timing logic.

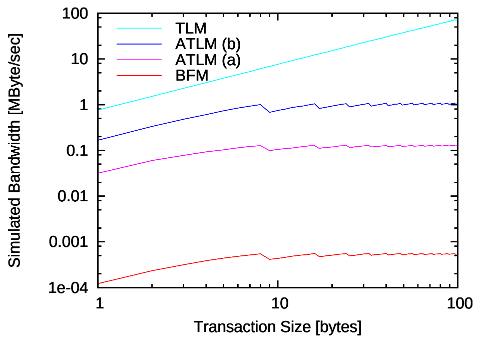


Fig. 16. Performance of the CAN models.

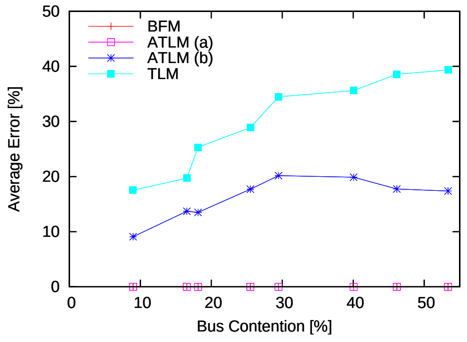
	BFM	ATLM (a)	ATLM (b)	TLM
Simulation Time [ms]	27.4	0.12	0.015	0.0012
Sim. Bandwidth [MByte/sec]	0.0006	0.127	1.05	12.3
Speedup over BFM	1	228	1879	22124
Rel. Speedup over previous	-	228	8	12

Table IV. Performance comparison for transferring 16 bytes using CAN models.

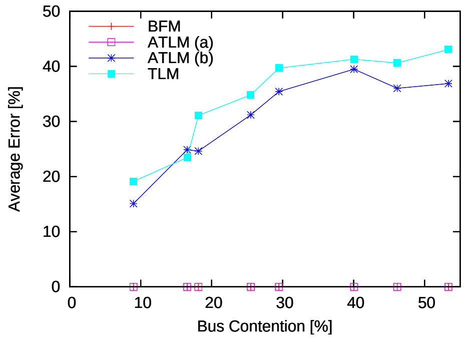
6.4 Accuracy Analysis

We use our generic dual master setup (Figure 6) for measuring the CAN model accuracy. However, CAN does not have a master/slave distinction, so we have modified our setup to two nodes acting as senders and two nodes acting as receivers. We let the user transactions vary in message id, length and content of the transaction (1 - 16 bytes), and in the delay between two transactions. Each sender uses an exclusive range of message identifiers. One will send messages with high priority identifiers (0-511), the other emits messages with low priority (identifiers 512-1023).

6.4.1 *Analysis Based on Individual Transfer Duration.* The first set of resulting graphs in Figure 17 show the average individual timing error over an increasing bus contention separately for high and low priority messages.



(a) high priority messages.



(b) low priority messages.

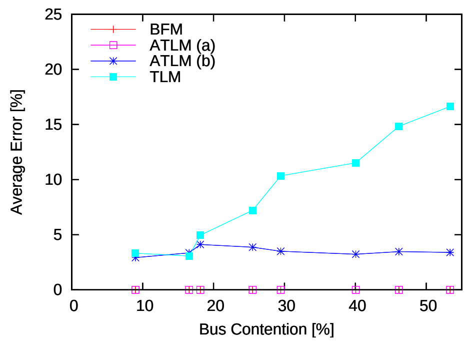
Fig. 17. Individual timing accuracy for the CAN models.

Figure 17(a) shows that the ATLM (a), which includes bit stuffing and CRC calculation, performs as accurate as the BFM (both graphs lie on top of the x-axis). This result has to be interpreted with perspective to the restrictions of the test, which are: no propagation delay between sending and receiving on the CAN bus, all delays between user transactions are multiple of the CAN bit time and the test starts aligned to the bit clock of the first sender. With this setting (reasonable for a simulation environment only) all bus accesses are performed aligned to the CAN bit clock, and no sub-cycle information is needed. In this situation, the additional capabilities of the BFM, i.e. bit synchronization, are not exercised and both the BFM and the ATLM (a) are accurate.

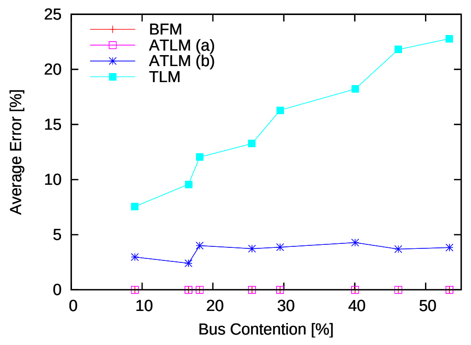
The ATLM (b), due to the missing bit stuffing and CRC, performs inaccurately. For messages in the high priority range, the inaccuracy starts with 10% for low contention situations and, after linearly rising to 20% inaccuracy, plateaus at 30% contention. Without the bit stuff modeling, an individual message transfer is - depending on its content - shorter than in the bus functional model. Therefore, the arbitration interaction between the two senders differs. With an increasing contention the user transactions of the low priority band increasingly influence the high priority transactions. However an earlier started low priority transaction, which may consist of multiple CAN frames, can delay a later started high priority user transaction only for up to one frame. A second started CAN frame of the low priority transaction will lose arbitration, which leads to the plateau in inaccuracy at 30%. Looking at the same scenario with reversed priorities, this limitation does not apply. A low priority user transaction may be delayed for a full high priority transaction consisting of many CAN frames. Hence, the timing error of the ATLM (b) increases without a plateau for the low priority user transactions (Figure 17(b)) with increasing contention.

The TLM yields uniform results for both the high and low priority sender. For both cases, the inaccuracy increases with the bus contention. As expected, the TLM suffers the most in loss of accuracy (40% inaccuracy at 45% contention).

6.4.2 *Analysis Based on Cumulative Transfer Duration.* Figure 18(a) and Figure 18(b) show the accuracy results based on cumulative transfer time.



(a) high priority messages.



(b) low priority messages

Fig. 18. Cumulative timing accuracy for the CAN models.

The analysis reveals that mispredictions of the ATLM (b) for individual CAN frames average out during the test, since the model correctly captures arbitration. Regardless of priority, a constant error of about 4% is measured. This can be attributed to not modeling the bit stuffing which in average adds 4% bits. The TLM, with its coarse grain priority-independent contention resolution, shows for both priority ranges a linear increasing error.

6.5 Summary for the CAN

Based on our analysis, Table V lists the fastest model that yields acceptable results for a given situation and simulation focus.

Environment Condition	**Applicable Model**
• No overlap between masters bus access • Early stage in design	TLM
• Main focus on application finish time	ATLM (b)
• Main focus on individual transfer delay	ATLM (a)
• Synthesizable • Using propagation delay	BFM

Table V. CAN model selection

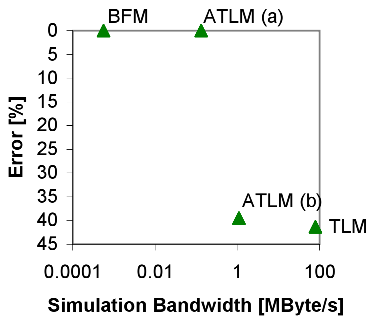


Fig. 19. CAN TLM trade-off.

The TLM can only be used in very early stages of the design. Its accuracy, for individual and cumulative transfer time, degrades drastically with increasing bus contention. The ATLM (b) is still fast. It is applicable in scenarios where the main focus is on the application finish time. However, this model is not suitable for predicting an individual transfer delay, since the duration-based analysis has not shown acceptable results.

The ATLM (a), which includes bit stuffing and CRC calculation, has shown 100% accuracy given the test restrictions (e.g. no propagation delay). It is the fastest model that accurately predicts the delay of an individual transfer in all contention situations. The BFM is necessary as a synthesizable model, or in case the simulation includes propagation delay on the simulated CAN bus.

Figure 19 depicts the TLM trade-off for the CAN using the average individual timing error of the lower priority node. The accurate models BFM and ATLM (a) are slow with a bandwidth of less than 1 MByte/s. The faster models yield inaccurate results, the ATLM (b) with 40% error. The fastest model, the TLM with close to 100 MByte/s, is also the most inaccurate with 43% error.

We classify the CAN as an example for an off-chip serial bus with decentralized arbitration. Its results can be used as an indicator for modeling other serial bus systems, such as I^{2}C [Philips 2000] and Ethernet.

7. COLDFIRE MASTER BUS

Our third bus example, the ColdFire Master Bus (Version 2.0) [Motorola 1997], falls into the category of custom CPU busses. Typically, those busses are simpler than general standard busses, since they have been designed for the specific requirements of one embedded processor.

7.1 Introduction

Motorola (now Freescale Semiconductor Inc.) has introduced the ColdFire Master Bus for its popular microprocessor ColdFire MCF5206. It is a 32 bit tristate bus that uses basic transfers (byte, short, word), as well as a 4 beat burst to fill or store a cache line. It has to be noted that the line access (like a 4-beat burst) cannot be preempted. Therefore, there is no distinction between locked and unlocked transfers, as we have seen for the AMBA AHB.

The ColdFire Master Bus allows the usage of two arbitration schemes. One, the three wire mode, involves an external arbiter and allows a true multi-master operation. The second option, the two wire mode, connects two masters that directly hand over the bus without the need of an external arbiter, as shown in Figure 20. We have selected the operation mode with two masters to complement the centralized arbitration scheme used for the AMBA AHB.

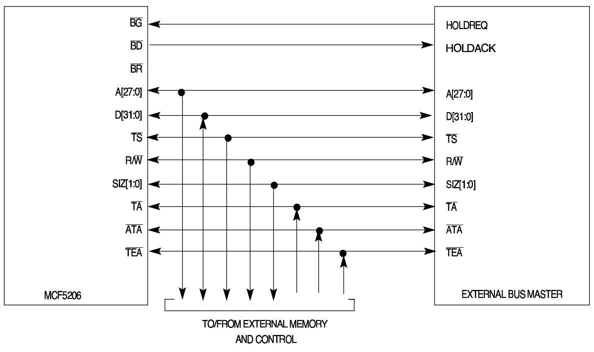


Fig. 20. ColdFire Master Bus with two masters (Source [Motorola 1997]).

7.2 Models

Our modeling of the ColdFire Master bus follows our scheme of abstracting data and arbitration, as described in Section 3. Again, we have implemented three major models: the BFM that implements a bus cycle access, the ATLM with a granularity of bus primitives, and the user transaction based TLM. We have not implemented any variants of these models.

7.3 Performance Analysis

We have measured the performance of the ColdFire Master bus in the setup with one master and one slave as described in Section 4.1. Figure 21 shows the simulation bandwidth.

Again, the measurements confirm the performance increase with abstraction. Also, the characteristic saw tooth shape, as seen already for the AHB, repeats. Due to the split of user transactions into bus transactions, the number of bus transactions does not increase linearly with an increase in size.

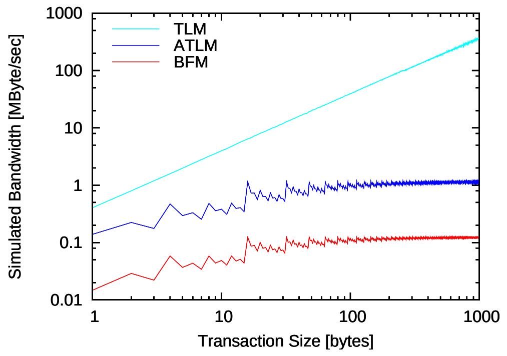


Fig. 21. Performance of the ColdFire Master bus models.

The BFM shows the lowest simulation bandwidth, since it models each individual bus cycle. The ATLM is one order of magnitude faster, due to the abstraction to bus transactions. The TLM is the fastest model. By simulating whole user transactions, it surpasses the ATLM by two orders of magnitude. Table VI lists detailed numerical results for transferring 512 bytes.

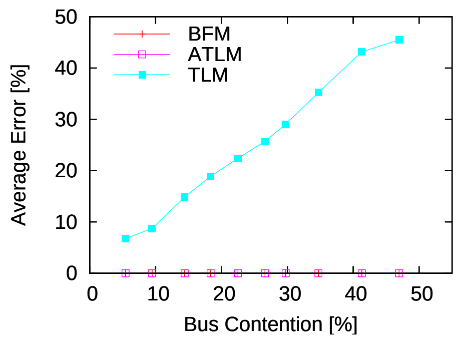
	BFM	**ATLM**	**TLM**
Simulation Time [ms]	3.93	0.423	0.0026
Sim. Bandwidth [MByte/s]	0.124	1.15	189.7
Speedup over BFM	1.0	9.3	1525.0
Rel. Speedup over previous	-	9.3	164.3

Table VI. Performance comparison for transferring 512 bytes using ColdFire Master bus models.

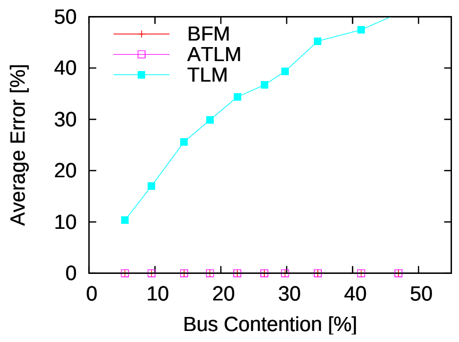
7.4 Accuracy Analysis

We have analyzed the timing accuracy in the setup with two concurrent masters as described in Section 4.2. As for the AHB, each master sends 5000 predefined user transactions that vary linear randomly in address, size (1 .. 100 bytes), and delay to the next transfer. Figure 22 depicts the accuracy of both masters over an increasing amount of bus contention.

As intended, the BFM complies with the standard and exhibits 0% error (its graph lies on top of the x-axis). The ATLM also reaches 100% accuracy. Since the ColdFire Master bus does not support preemption, the arbitration handling at



(a) High priority master.



(b) Low priority master.

Fig. 22. Individual timing accuracy for the ColdFire Master bus models.

the boundary of the bus transaction is sufficient. In case the high priority master requests the bus during an ongoing burst of the low priority master, the high priority master has to wait until the burst completes and the low priority master releases the bus. The bus ownership changes only at the boundary of a bus transaction. Thus, the ATLM, which models this granularity, is accurate and its graph lies on top of the x-axis.

The TLM exhibits a linear increase in error with increasing bus contention, since it models the transfers at the coarser granularity of user transactions. At 45% bus contention, the TLM shows 54% error for the low priority master.

Analyzing the cumulative transfer times yields very similar results, so we omit these graphs. The TLM is the only model that exhibits an error, which reaches 35% at 45% bus contention.

7.5 Summary for the ColdFire Master Bus

With the Motorola ColdFire Master bus being simpler than the other protocols, its speedup with abstraction is not as dramatic. The most abstract TLM is 1525 times faster than the BFM. At the same time, the TLM is the only model that produces inaccurate results, with up to 54% error. The ATLM already simulates accurately, since the bus does not support burst preemption. However, the ATLM is only 9.3 times faster than the BFM.

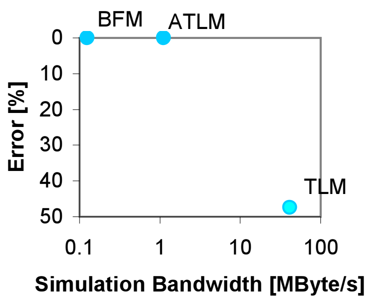


Fig. 23. ColdFire Master bus TLM trade-off.

Figure 23 shows the trade-off for the ColdFire Master bus. The accurate models BFM and ATLM are slow with a bandwidth below 1 MByte/s. The fast TLM, on the other hand, only yields inaccurate results with an average 47% error.

8. GENERALIZATION

Combining the performance and accuracy results of the three modeled bus systems, we will now generalize this data to give a broader perspective on the benefits and drawbacks of TLM. While, for an existing multitude of embedded bus architectures, an extrapolation based on three individual models is difficult in general, our data is a strong indication of generality.

As basis for our generalization, we have carefully selected three standard examples found in a majority of real-world embedded systems that exhibit very diverse communication properties. The AMBA AHB is a general-purpose on-chip pipelined parallel bus with central arbitration and a multiplexed interconnection scheme. In contrast, the CAN is a dedicated off-chip serial bus with distributed arbitration and a dominant-recessive interconnection. Yet a different candidate is the Motorola ColdFire Master Bus, a special-purpose embedded processor bus with custom arbitration and tri-state interconnection. Thus, these three chosen busses cover a wide variety of bus categories and can therefore serve as basis for a generalization with strong confidence.

Moreover, our granularity-based modeling approach is general and applicable to any layer-based bus model. Finally, we have observed the very same performance and accuracy patterns in our analysis for all three cases.

8.1 Performance

All our measurements confirm that TLM abstraction dramatically increases simulation performance (see Table VII). Decreasing granularity of user data handling and arbitration is a very efficient method of abstraction. For all models, we have measured a speedup of orders of magnitude with each step of coarser granularity. The cycle-accurate BFM performs slowly for all busses. The ATLM, that simulates on a bus transaction granularity, is at least one order of magnitude faster. Further decreasing the granularity to user transactions, as done by the TLM, increases the speed by another two orders of magnitude.

Bus	**ATLM (a)**	**ATLM (b)**	**TLM**
AMBA AHB	78	81	6802
CAN	228	1879	22124
ColdFire	9.3	-	1525

Table VII. Speedup over bus functional model.

On the other hand, variants at the same granularity level do not yield any significant speedup. We see this clearly in the ATLM variants for the AMBA AHB. In other words, models within the same granularity category simulate at the same speed. It should be noted that we have observed the same fact also for variants at the TLM abstraction in previous work [Schirner and Dömer 2006].

Only, if a feature requires a significant computation effort, then abstracting even at the same granularity level can improve performance. As an example, the ATLM (b) of the CAN does not model bit inspection and simulates 8 times faster than its counterpart at the same level.

Our analysis also shows that the potential performance gain for a model depends on the complexity of the actual protocol. We have achieved the highest speedup of 22124x over the BFM for the complex CAN protocol. On the other hand, for the relative simple protocol of the custom CPU bus, the Motorola ColdFire Master bus, the TLM is only 1525 times faster.

8.2 Accuracy

Table VIII summarizes the results of our accuracy analysis. The accuracy achieved by a model depends on the modeling granularity and the actual granularity of the bus. By definition, each BFM is accurate, since it is the most fine-grained model. The ATLMs based on bus transactions are accurate if (and only if) the modeling granularity matches the actual granularity of the bus. The ATLM (a) for the AHB was accurate in the locked transfer mode as well as the ATLMs for the CAN and the ColdFire Master bus. Here, the actual bus protocol arbitrates once per bus transaction.

Bus	**BFM**	**ATLM (a)**	**ATLM (b)**	**TLM**
AMBA AHB (locked)	0%	0%	18%	32%
AMBA AHB (unlocked)	0%	31%	31%	37%
CAN	0%	0%	39%	42%
ColdFire	0%	0%	-	47%

Table VIII. Average individual timing error for the low priority master at 40% bus contention.

Inaccuracy has to be accepted when the model is more coarse grained. The ATLM (a) is inaccurate in the unlocked mode of the AMBA AHB, since bursts can be preempted and an arbitration decision within a bus transaction would be necessary. Furthermore, all our TLMs are inaccurate, due to the high abstraction.

Abstracting away timing relevant features also leads to an inaccurate model, as shown for the CAN ATLM (b) that omits CRC calculation and bit stuffing. Also, the abbreviated arbitration of the ATLM (b) of the AMBA AHB yields an inaccurate model.

Finally, the actual measured inaccuracy of a model highly depends on the bus contention. The error of coarse grain arbitration modeling only takes effect if two masters access the bus simultaneously. In other words, all our models are accurate if there is no bus contention. Therefore, the most abstract model is the best choice if the architecture at hand only uses a single master, or if a very low bus contention can be expected. On the same note, modeling point-to-point dedicated links that inherently do not have any bus contention, can also be accurately modeled at the highest TLM abstraction.

8.3 TLM Trade-Off

Figure 24 summarizes our measurements and depicts the TLM trade-off for all our bus models. On the x-axis, Figure 24 denotes the simulation performance in terms of the simulation time for a 100 byte user transaction. The y-axis denotes the average error of the model for the low priority master at 40% bus contention.

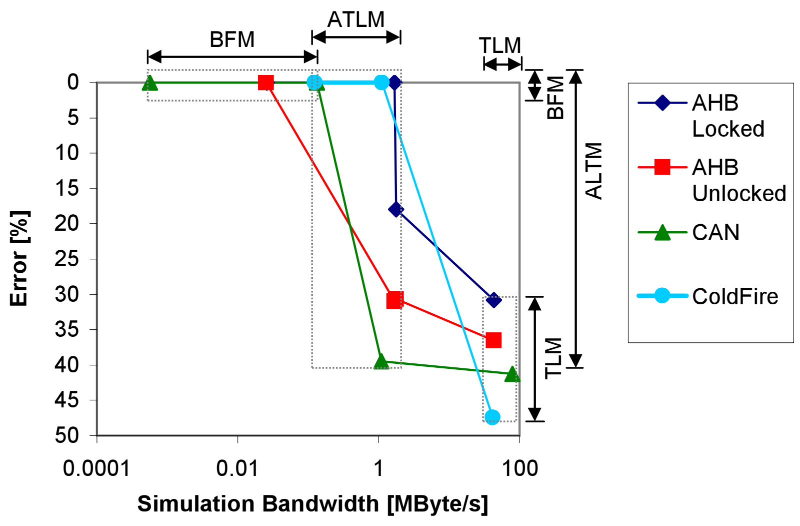


Fig. 24. TLM trade-off summary.

Abstract modeling poses a trade-off, with either fast or accurate results. The accurate BFMs are slow with less than 0.2 MByte/s bandwidth. On the other hand, the fast TLMs with a bandwidth of up to 100 MByte/s produce an average error from 32% to 47%.

The ATLMs are found in the middle, simulating at about 1 MByte/s bandwidth. Some are accurate, since the modeled granularity matches the granularity of the actual protocol. If the granularity does not match, or in case of additional feature abstraction, the ATLMs generate an average error from 18% to 39%.

9. CONCLUSIONS

Our contributions in this article are fourfold. First, we have classified TLM according to the granularity of data and arbitration handling. Second, we have defined appropriate metrics and setups for a systematic quantitative analysis of TLM with respect to simulation performance and timing accuracy. Third, we have quantified the TLM trade-off for three diverse communication protocols. Fourth, we have generalized our observations from the analysis to guide communication model designers and model users.

In particular, we have applied our granularity-based abstraction to three common bus systems covering diverse communication protocols: first the AMBA AHB, as an on-chip parallel bus with centralized arbitration, second the CAN, as an off-chip decentralized serial bus system, and third the ColdFire Master bus, as a custom embedded processor bus. We have modeled, validated, and systematically analyzed each bus using our performance and accuracy metrics.

Based on the analysis results of the individual examples, we have then derived general conclusions. Abstraction based on a decreasing (coarsening) granularity yields at least an order of magnitude improvement per granularity level. On the other hand, abstracting features at the same granularity level only yields marginal performance improvements. In other words, proper model granularity is the key to efficient TLM abstraction.

However, TLM abstraction results in a serious loss in accuracy if the modeled granularity is more coarse grain than the granularity present in the actual bus protocol. This defines the TLM trade-off. In general, a model is either fast or accurate. Our fast TLM models with up to 100 MBytes/s bandwidth show an error of up to 47%. Accurate models, on the other hand, are slow. Our BFMs simulate with less than 0.2 MByte/s bandwidth.

The actual measured timing error highly depends on the bus contention. All our models are accurate in the absence of bus contention. Therefore, in the special case of a single master architecture, the most abstract TLM can be used without loss of accuracy.

In conclusion, this article contributes a systematic quantitative analysis of performance and accuracy in TLM, using a diverse set of major bus architecture standards, that confirms the TLM promise of high simulation speed. Our detailed analysis also identifies conditions for abstract and accurate models. As a result, we provide guidelines to the designer of communication models for efficiently abstracting communication protocols. The same guidelines also allow the system designer as a user of communication models to make an informed decision about the appropriate model for her/his design at hand.

REFERENCES

ARM. 1999. *AMBA Specification (Rev. 2.0), ARM IHI 0011A.* Advanced RISC Machines Ltd. (ARM).

ARM. 2003. *AMBA AHB Cycle Level Interface (AHB CLI) Specification, ARM IHI 0011A.* Advanced RISC Machines Ltd (ARM).

Bosch. 1991. *CAN Specification*, 2.0 ed. Robert Bosch GmbH.

Brem, D. and Müller, D. 2003. Interface based system modeling of a can using sve. In *Proceedings of the EkompaSS Workshop.* Hanover, Germany.

Cai, L. and Gajski, D. 2003. Transaction Level Modeling: An Overview. In *Proceedings of the International Conference on Hardware/Software Codesign and System Synthesis.* Newport Beach, CA.

Caldari, M., Conti, M., Coppola, M., Curaba, S., Pieralisi, L., and Turchetti, C. 2003. Transaction-level models for AMBA bus architecture using SystemC 2.0. In *Proceedings of the Design, Automation and Test in Europe (DATE) Conference.* Munich, Germany.

Coppola, M., Curaba, S., Grammatikakis, M., and Maruccia, G. 2003. IPSIM: SystemC 3.0 enhancements for communication refinement. In *Proceedings of the Design, Automation and Test in Europe (DATE) Conference.* Munich, Germany.

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

Gasteier, M. and Glesner, M. 1996. Bus-Based Communication Synthesis on System-Level. In *Proceedings of the International Symposium on System Synthesis.* San Diego, CA, USA.

Gerstlauer, A. and Gajski, D. D. 2002. System-level abstraction semantics. In *Proceedings of the International Symposium on System Synthesis.* Kyoto, Japan.

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

Gerstlauer, A., Shin, D., Peng, J., Dömer, R., and Gajski, D. D. 2007. Automatic Layer-Based Generation of System-On-Chip Bus Communication Models. *IEEE Transactions on Computer-Aided Design of Intergrated Circuits and Systems (TCAD) 26,* 9, 1676–1687.

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

Hartwich, F. and Bassemir, A. 1999. *The Configuration of the CAN Bit Timing.* Robert Bosch GmbH. http://www.can.bosch.com.

Haverinen, A., Leclercq, M., Weyrich, N., and Wingard, D. 2002. SystemC based SoC Communication Modeling for the OCP Protocol. http://www.ocpip.org.

IBM. 2004. *128-bit Processor Local Bus Architecture Specification, SA-14-2538-04*, 4.6 ed. IBM.

ISO. 1994. *Reference Model of Open System Interconnection (OSI)*, Second ed. Internation Organization for Standardization (ISO). ISO/IEC 7498 Standard.

Lahiri, K., Raghunathan, A., and Dey, S. 2001. System-Level Performance Analysis for Designing On-Chip Communication Architectures. In *IEEE Transactions on Computer-Aided Design of Intergrated Circuits and Systems (TCAD).* Vol. 20. 768–783.

Lajolo, M., Passerone, C., and Lavagno, L. 2003. Scalable Techniques for System-level Co-Simulation and Co-Estimation. In *IEE Proceedings–Computers and Digital Techniques.* Vol. 150. 227–238.

Motorola 1997. *MCF5206 ColdFire Integrated Microprocessor User's Manual.* Motorola.

Pasricha, S., Dutt, N., and Ben-Romdhane, M. 2004. Fast Exploration of Bus-based On-chip Communication Architectures. In *CODES and ISSS.* Stockholm, Sweden.

Philips. *P8xC592: 8-bit microcontroller with on-chip CAN.* Philips.

Philips 2000. *The I^2C-bus specification*, 2.1 ed. Philips.

Rose, A., Swan, S., Pierce, J., and Fernandez, J.-M. 2005. Transaction Level Modeling in SystemC. http://www.systemc.org.

Sarmento, A., Cesario, W., and Jerraya, A. A. 2001. Mixed-Level Cosimulation for Fine Gradual Refinement of Communication in SoC Design. In *Proceedings of the Design, Automation and Test in Europe (DATE) Conference.* Munich, Germany.

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

Schirner, G. and Dömer, R. 2005b. System Level Modeling of an AMBA Bus. Tech. Rep. CECS-TR-05-03, Center for Embedded Computer Systems, University of California, Irvine. March.

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

Sgroi, M., Sheets, M., Mihal, M., Keutzer, K., Malik, S., Rabaey, J., , and Sangiovanni-Vincentelli, A. 2001. Addressing the System-on-a-Chip interconnect woes through communication based design. In *Proceedings of the Design Automation Conference.* Las Vegas, NV, USA.

Siegmund, R. and Müller, D. 2001. 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.

Received: June 2006; revised February 2007; accepted August 2007