Accuracy-Adaptive Simulation of Transaction Level Models

M. Radetzki, R. Salimi Khaligh
Institut für Technische Informatik, Universität Stuttgart, Stuttgart, Germany
{radetzki, salimi}@informatik.uni-stuttgart.de

Abstract

Simulation of transaction level models (TLMs) is an established embedded systems design technique. Its use cases include virtual prototyping for early software development, platform simulation for design space exploration, and reference modelling for verification. The different use cases mandate different trade-offs between simulation performance and accuracy. Therefore, multiple TLM abstraction layers have been defined of which one has to be chosen and integrated into the system model prior to simulation. In this contribution we present a modelling technique that allows covering several layers in a single model and switching between the layers at any time, in particular dynamically during simulation. This feature is employed to automatically adapt simulation accuracy to an appropriate level depending on the model's state, leading to an improved trade-off between simulation performance and accuracy.

1 Introduction

Transaction level modelling aims at enabling system level simulation of large systems, for which RTL simulation would require an unacceptable amount of time. This goal is achieved by abstracting from signal level communication and modelling complex communication operations as atomic transactions, thereby reducing the number of events to be processed by event-driven simulators. In the case of bus based platforms, transactions model bus transfers such as burst or single word read and write operations.

Transaction level models exist at different layers which can roughly be characterized as untimed, approximately-timed, cycle-approximate and cycle-accurate. These layers provide different trade-offs between abstraction and accuracy suitable for different use models.

The decision which layer to employ is made by the user prior to simulation based on the use case and required accuracy. Hence, the same layer is employed throughout a complete simulation run. This layer may, however, not at all times during the simulation constitute an optimal performance / accuracy trade-off. For example, assume that a model with pre-emption of bus transfers is being simulated in order to perform bus performance estimation with relatively high accuracy. During simulation periods in which only one master or the highest priority master is active, a more abstract layer without pre-emption could be employed to gain simulation speed at no loss of accuracy. On the other hand, when a situation occurs that requires simulation on a more accurate layer, a temporary switch to this layer can help increase accuracy at little performance cost (as long as it occurs infrequently).

In the remainder of this paper, we present a novel approach that collects different layers in a single model and automatically adapts the accuracy (or layer) to the changing simulation scenarios. Related work is analyzed in the next section. Section 3 outlines the fundamental concepts of adaptive TLMs, and section 4 provides details on a SystemC based implementation. In section 5, we present experimental results on simulation performance. Section 6 concludes our contribution.

2 Related work

Many TLM layers and terminologies have been suggested by multiple authors and groups [2]. We adopt the terminology of the SystemC TLM group for our work but give a more precise definition of the layers based on Niemann's and Haubelt's concept of atomic transactions [4]: At the PV and PV+T layers, a bus transfer is modelled as a single transaction; at CA layer, a transfer is modelled with at least one transaction per bus cycle, and at CX layer(s) with more transactions than PV(+T) but less than CA.

TLM layers are typically separated into disjoint models and few models describe a relation between the layers. The GreenBus [3] is a generic bus model that introduces the concept of transactions consisting of uninterruptible phases (atoms) which are collections of payload values (quarks). By identifying bus protocol signals as quarks, the generic model can be customized to model concrete buses. Simulation can be performed at the layer of transactions, atoms or quarks, corresponding to approximately-timed, cycle-approximate, and cycle-accurate layers.

Multi-accuracy simulation has been introduced to TLM by Beltrame et al. [1] for the purpose of switching at run time between power models with different accuracy / performance trade-offs. Accuracy selection is performed under user control by multiplexing and demultiplexing different

978-3-9810801-3-1/DATE08 © 2008 EDAA

channel models. Switching is not instantaneous since transactions in the switched-off model need to be completed while new transactions are initiated with the switched-on model. This appears acceptable for the main application of simulating a model with low accuracy until a point of interest is reached at which switching to a higher accuracy model can be done. However, it is not possible to react quickly to simulation scenarios such as transfer pre-emption.

A modelling approach adaptive to simulation scenarios has been presented by Schirner and Dömer [7] under the term Result Oriented Modeling (ROM). ROM simulates a complete bus transfer as an atomic transaction and extends the duration of a transaction when the bus transfer would have been pre-empted in the real system. Based on the authors' definition of timing accuracy, which only considers the simulated start and end points of each transaction, the model is fully timing accurate. However, all data is transferred at the beginning of a transaction. Hence, with respect to data, the model is not cycle-accurate and functional correctness can be impaired. E.g., if a write transfer to a RAM slave is pre-empted by a read transfer, a data value may be read that has not yet been transferred in the real system.

3 Adaptive modelling

Cycle-accurate TLMs perform bus arbitration in each bus cycle. We reduce the amount of arbitration processing under the assumption of a time-invariant arbitration protocol: Since the grant decision is not modified unless the state of the waiting and active transfers changes, arbitration can be limited to cycles in which a new communication request arrives to the bus or in which the currently active transaction releases the bus (e.g. because the transfer is finished), giving waiting masters the chance to be granted the bus. The need for arbitration is signalled by a single, central event in our bus model. Figure 1 depicts the time instants at which this event has to be triggered. In the best case (performance-wise), this reduces arbitration effort significantly compared to the cycle-accurate approach. On the other hand, in cases of high bus utilization with transfers being initiated or finished in each cycle, the model inherently converges towards cycle-accuracy.

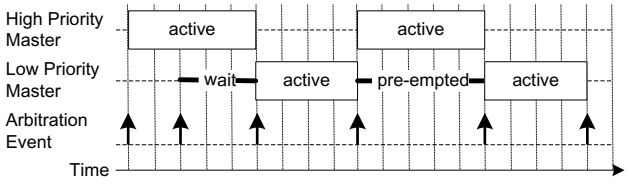

Fig. 1: Triggering the arbitration event

As a result of bus arbitration, a transfer may be active without pre-emption from its initiation until completion. Such a transfer can be modelled accurately and efficiently as a single atomic transaction. Whether this is the case depends on bus requests by other masters (of potentially higher priority) while the transfer is active. This information, however, is not known before the transfer is finished. Therefore, data is transferred as a single block at the end of a transaction in our model as shown in Figure 2. In absence of interfering transfers, the end time of a transaction can precisely be estimated from its start time, the amount of data to be transferred and the bus cycle time.

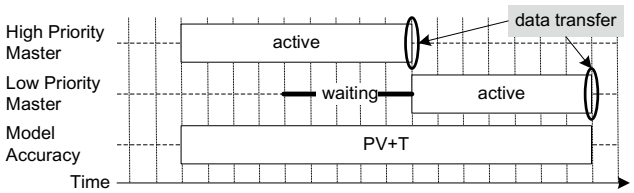

Fig. 2: Uninterrupted transfers as single transactions

During the simulation of a bus transfer, another transfer may be initiated by a higher-priority master. Then, after arbitration, the current bus transfer must be pre-empted. An atomic transaction in this case only models an uninterrupted portion of the transfer. Simulation adapts to this scenario by quitting the transaction and transferring an amount of data corresponding to transaction duration as shown in Figure 3. The state of the transfer is stored for later resumption. Then, the higher-priority transfer is simulated using the same adaptive approach, i.e. starting with the ability to cover the complete transfer efficiently and accurately with a single transaction, yet ready to partition it into multiple transactions should the simulation scenario require it.

After completion of the high-priority transfer, simulation resumes the pre-empted transfer with a new transaction which starts with the stored transfer state. The transfer may be pre-empted again, which is handled as explained before. Otherwise, the simulation of the transfer is finalized with this transaction, and only the remaining amount of data is transferred at its end.

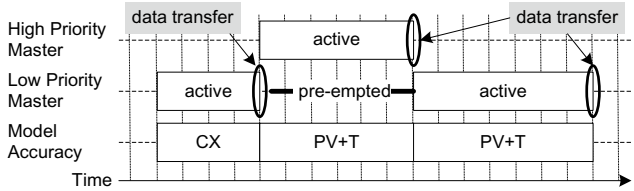

Fig. 3: Pre-empted transfer as multiple transactions

The simulation mechanism automatically adapts to cycle-accuracy if bus transfers are initiated or completed in every cycle. In this case, bus arbitration is performed cycle-by-cycle and transaction duration may be reduced to a single cycle in the extreme case. However, if arbitration does not result in pre-emption, transactions can still extend over more than a cycle without losing accuracy, thereby achieving higher simulation performance than a purely cycle-accurate approach.

4 Implementation in SystemC

We have implemented the adaptive TLM concepts in a SystemC model based on the object-oriented approach of [5]. The API for initiating transfers is provided by a master port, a specialized SystemC port which implements operations for performing transfers over the bus (cf. Figure 4). A transfer is initiated by calling the *read* or *write* operation, respectively. The base address to be accessed and the number of bus data words to be transmitted are passed via the parameters *addr* and *size*. If *size* is greater than one, the transfer is a burst to successive addresses starting from *addr*. The data of a write transfer must be supplied by calls to the operation *put*, and read data can be obtained via *get*. This facilitates cycle-true modelling of the master: it can initiate, e.g., a write transfer without having to know all data in advance, and can later provide the data words as they are computed cycle by cycle. On the other hand, the master may abstract from timing by supplying all data without delay just after initiating the transfer, and at any time synchronize to the timed bus model by calling the *finish* operation which waits for transfer completion.

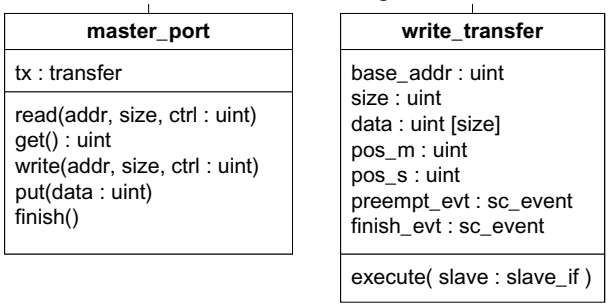

Fig. 4: Transfer API and data structure

When a transfer is initiated, a corresponding transfer message is created and stored in the port's attribute *tx*. Figure 4 shows the message that represents a write transfer. In addition to address, burst size and data, the message carries information on the amount of data, *pos_m*, that has been supplied by the master via *put* and the position in the burst, *pos_s*, that is to be processed next by the slave. Each transfer object includes two events, *preempt_evt* and *finish_evt*, that are used by the bus model to signal the preemption and completion of a transfer, respectively.

Figure 5 is a UML activity diagram specifying the protocol-specific behaviour of the bus model, where

- *m* is the master that currently owns the bus,
- *m'* is the master to which the bus is handed over,
- *tx* is a reference to the current transfer,
- *s* is the slave addressed by the transfer, and
- *put*, *get*, and *peek* are operations of TLM channels as defined by the SystemC TLM standard.

The behaviour consists of the phases (1)-(4) shown in Fig. 5. It is triggered (1) by a central event, *arbitrate_evt*, which occurs as described in Section 3. In phase 2, the current transfer (if any) is removed from the bus in case it is finished. In phase 3, the new master that gets the bus according to the modelled arbitration scheme is determined; this may, e.g. in a round robin scheme, depend on the current master. The function *arbitrate* shall skip masters that have no transfer message. Finally, in phase 4, the transfer message of the new master (if any) is forwarded to the addressed slave after pre-emption of a previously active transfer (if any).

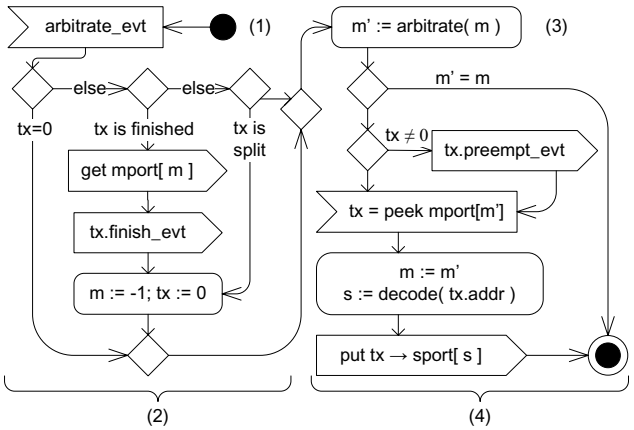

Fig. 5: Arbitration processing activity

The behaviour explained so far is independent of simulation accuracy. Layer-dependent and adaptive behaviour are assembled in the transaction model as follows:

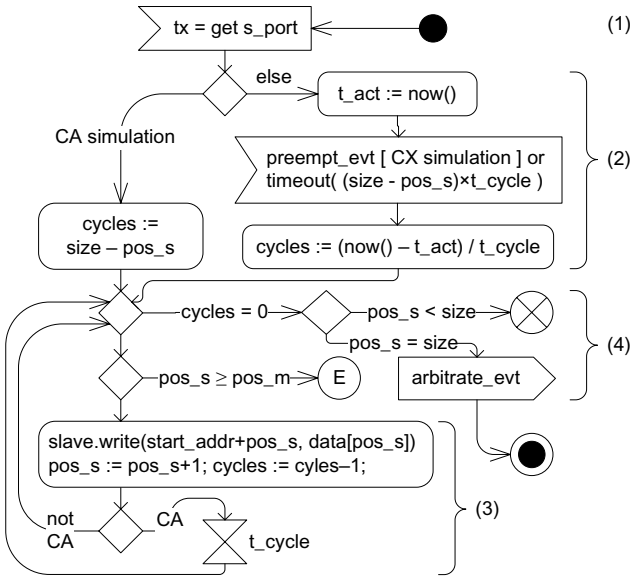

Fig. 6: Transaction processing activity

The transfer's *execute* method implements the slave side of processing a transfer as illustrated in Figure 6 with the example of a write transfer. The activity begins when a transaction is received (1) over the slave's port. The activity waits (2) for the maximum duration of the transaction (timeout), corresponding to PV+T layer simulation. A pre-emp-

tion by the bus model may occur if CX simulation is enabled. At the end of phase 2, the number of bus cycles elapsed while waiting is computed. A value less than the full transfer time represents the case of cycle-approximate simulation.

The slave operations that would have taken place during the elapsed bus cycles in the modelled system are performed retroactively in phase 3. If the slave would proceed beyond the amount of data put by the master so far (*pos_s* reaches *pos_m*), an error occurs (E). Alternatively, wait operations could be introduced to model busy cycles (or slave wait states). The completion of a transaction is shown in phase (4): If the end of the transfer is reached, the arbitration event is released. Otherwise, the transfer will be completed later by starting the activity again.

5 Experimental results

The experimental setup used for performance evaluation of the adaptive simulation approach has been chosen so as to enable a comparison with the related work [7]. Simulations have been performed on a computer with Pentium M 1.66 GHz. The setup includes two masters and one slave. The first master issues transfers of increasing burst length. The second master has a higher priority and issues transfer requests of eight-beat bursts at a constant rate to achieve a base utilization u of 10 to 50 percent. With increasing u, an increasing pre-emption of the first master's transfers has to be simulated.

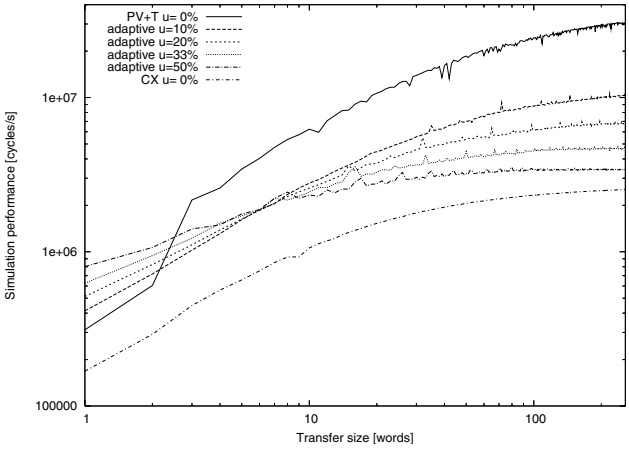

Fig. 7: Performance for varying amount of pre-emption

Figure 7 shows the resulting simulation performance of the adaptive model in comparison to pure PV+T [5] and CX [6] models. Since our model switches between PV+T and CX accuracy, its performance is in-between the performance of the fixed-layer models. With higher base utilization, performance decreases because more pre-emptions have to be simulated, and simulation accuracy is more often at CX layer. However, for transfer sizes less than 8 words, this relationship is reversed for two reasons:

- Shorter low-priority transfers are more likely to be finished just in time when a high-priority transfer arrives to the bus, so that no pre-emption occurs and simulation can be at the more efficient PV+T layer.
- The high-priority transfers are never pre-empted and due to their longer duration simulated more efficiently than the shorter low-priority transfers. This is also the reason for exceeding PV+T performance (measured with $u = 0\%$) at transfer size less than three words.

Performance of the ROM approach has been measured in [7] as the simulation time for a transfer at $u = 33\%$ and ranges from 0.002 to 0.05 milliseconds for transfer sizes from 1 to 1024 bytes. Corresponding measurements with our model show simulation times from 0.003 to 0.08 milliseconds. Hence, adaptive modelling can remove the intra-transaction inaccuracy inherent to ROM at a slight performance penalty.

6 Conclusions

We have introduced a new, accuracy-adaptive transaction level modelling and simulation concept that relieves the user of handling multiple models at different abstraction layers. The concept has been implemented in a model that achieves cycle-by-cycle data accuracy if required for functional correctness but switches to more abstract simulation of transfers at completion and pre-emption points when possible. Simulation performance is significantly above previous CA and CX models and comes close to the PV+T layer of non-preemptible transfers.

7 References

[1] G. Beltrame, D. Sciuto, C. Silvano: *Multi-Accuracy Power and Performance Transaction-Level Modeling*. IEEE Trans. on Computer Aided Design of Integrated Circuits and Systems (TCAD), vol. 26, pp. 1830-1842, 2007.

[2] M. Burton at al.: *Transaction Level Modelling: A Reflection on What TLM Is and How TLMs May Be Classified*. Proc. Forum on Design Languages (FDL), 2007.

[3] W. Klingauf et al.: *GreenBus - A Generic Interconnect Fabric for Transaction Level Modelling*. Proc. 43rd Design Automation Conference (DAC), 2006.

[4] B. Niemann, Ch. Haubelt: *Towards a Unified Execution Model for Transactions in TLM*. Proc. Formal Methods and Models for Codesign (MEMOCODE), 2007.

[5] M. Radetzki: *SystemC TLM Transaction Modelling and Dispatch for Active Objects*. Proc. Forum on Design Languages (FDL), 2006.

[6] M. Radetzki, R. Salimi Khaligh: *Modelling Alternatives for Cycle Approximate Bus TLMs*. Proc. Forum on Design Languages (FDL), 2007.

[7] G. Schirner, R. Dömer: *Fast and Accurate Transaction Level Models using Result Oriented Modeling*. Proc. Int'l Conference on Computer Aided Design (ICCAD), 2006.