LETTER

Scenario-Aware Bus Functional Modeling for Architecture-Level Performance Analysis

Eui-Young CHUNG[†], ***Member***, **Hyuk-Jun LEE**[††], ***and*** **Sung Woo CHUNG**[†††a)], ***Nonmembers***

SUMMARY We present a scenario-aware bus functional modeling method which improves the accuracy of traditional methods without sacrificing the simulation run time. Existing methods focused on the behavior of individual IP (Intellectual Property) components and neglected the interplay effects among them, resulting in accuracy degradation from the system perspective. On the other hand, our method thoroughly considers such effects and increases the analysis accuracy by adopting control signal modeling and hierarchical stochastic modeling. Furthermore, our method minimizes the additional design time by reusing the simulation results of each IP component and an automated design flow. The experimental results show that the accuracy of our method is over 90% of RTL simulation in a multimedia SoC (System-on-Chip) design.

key words: *bus functional model, bus, stochastic, performance, inter-play*

1. Introduction

Reducing the time-to-market is a critical issue in SoC (System-on-Chip) design due to the rapid growth of design complexity. A viable solution is a platform-based design by reusing existing IP components [1], [2]. One of the issues in platform-based design is the on-chip communication architecture design to meet the bandwidth and latency constraints required by the IP components consisting of the platform [2]–[4]. Traditional RTL simulation is too slow to be used for this purpose and several approaches have been proposed instead. Bus Functional Modeling (BFM) simply models the traffic pattern of each IP component in transaction-level and connects them to communication architectures for several types of performance analysis [4]. The trace-driven model is widely used for its simplicity, but it requires a large disk space to maintain its database [4]. Furthermore, it is not flexible enough to analyze the performance of a SoC for various operation modes, since the trace only reflects a special operation mode that an IP component can perform. Another approach is using a stochastic model based on the probabilities of traffic types [5]. This approach is more flexible than the trace-driven method, but its accuracy is lower than the trace-driven method for a specific operation mode. More importantly, these approaches pay little attention on the inter-play effect of IP components in BFM based performance analysis.

Manuscript received June 26, 2006.
Manuscript revised October 4, 2006.
Final manuscript received January 4, 2007.
[†]The author is with Yonsei University, Seoul, Korea.
[††]The author is with Cisco Systems Incorporation, CA, USA.
[†††]The author is with Korea University, Seoul, Korea, Corresponding author.
a) E-mail: swchung@korea.ac.kr
DOI: 10.1093/ietfec/e90–a.4.875

Since some IP components are closely related due to their nature, the accuracy of overall system performance analysis critically depends on their interactions. Moreover, an IP component can behave in several different ways. For example, a multi-format video decoder will produce different traffic patterns depending on the decoded video format. In this case, each video format will be the basic unit to define the IP component behavior. Also, we call each different behavior of an IP component operation mode. It is typical that a controller determines the operation mode and operation start time of each IP component. It is necessary to model them for exact performance analysis. Another example can be found when private signals of an IP component initiate the operation of other IP components for the same reason. One promising method to attack this issue is System-C based transaction-level modeling [2], [3]. In this method, the interface with on-chip communication fabrics including bus is modeled in a cycle-accurate or cycle-approximate manner. The accuracy of simulation is achieved at the cost of simulation speed. For accurate performance analysis, we need to model each IP component in micro-architecture level with reasonable timing information. Such requirement increases the modeling effort and thus its practical use is limited.

The method proposed in this paper improves the shortcomings of traditional BFM methods by considering the inter-play effect among IP components, but our method does not require a large amount of modeling effort by reusing the simulation results collected during the verification of each IP. Each BFM includes several traffic generators corresponding to various possible operation modes. An operation mode is dynamically selected by other BFMs or designer specification to reflect the operation scenario intended by designers, thus the BFMs in our method model not only bus transactions, but also control signals to capture the inter-dependency of IP components. The traffic generator for each operation mode can be modeled using a trace-driven method or a stochastic method depending on the designer's choice. In the stochastic method, we maximize its accuracy by extracting the state transition diagram from the simulation trace. Each line in the trace corresponds to each transaction and the states are created by grouping the trace lines based on their interval (transaction interval) and burst length. Each state generates traffic patterns based on these two parameters and the transition probability from state A to state B is the state transition count from state A to state B over the total transition count from state A when we replace

the trace lines by the mapping states.

The method proposed in this paper does not count on a specific bus protocol, nor a specific modeling language. We use AMBA AHB from ARM and Verilog HDL as an exemplary case. In Sect. 2, we will describe the details of our method and provide the experimental results to show its effectiveness in Sect. 3 followed by a conclusion in Sect. 4.

2. Scenario-Aware Bus Functional Model

2.1 Bus Functional Model Structure

The IP component consists of two parts — computation and communication as shown in Fig. 1.

The computation part processes are in charge of data processing and communication part transfers data through the communication fabric. Communication part is common to every BFM for a given bus protocol. To hide the details of signaling, we adopt transaction-level modeling and only the high level abstraction is visible to the computation part. Computation part includes a series of read/write transactions which are delivered to the communication part. The final outputs from communication part are traffic patterns similar to those from RTL IP component. In addition to the signals specified by a given bus protocol, we introduce virtual control signals to reflect the inter-dependency of BFMs. The control signals are fed from other masters, slaves, or designer specification. Also, some control signals are going out if necessary. With these control signals, we can model various operation scenarios which are critical from a performance perspective.

2.2 Control Signals for Bus Functional Model

Figure 2 shows an example of BFM behavior depicted in Fig. 1. In this example, we introduce three types of control signals.

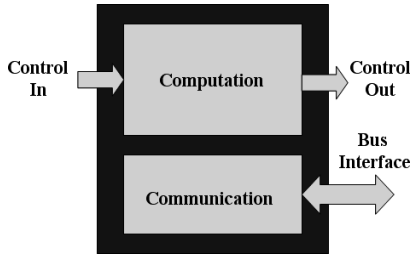


Fig. 1 BFM structure.

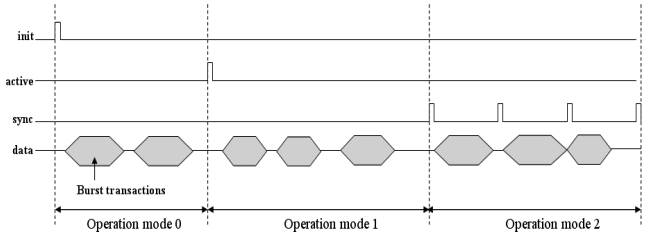


Fig. 2 An example of BFM behavior.

The first type of control signal, "init" triggers the start of BFM and the triggering time is specified by a designer in a configuration file. Then, the BFM generates traffic patterns defined by the computation part which can be a trace or a stochastic model extracted from RTL simulation trace. After completing the pattern generation, BFM stays in idle state and waits for other control signals. Next, the second type of control signal "active" triggers BFM and then it generates traffic patterns again. Note that the traffic pattern at this time is different from the previous one, since BFM performs a different operation specified by the control signal. Thus, BFM may have multiple pattern generators in computation part and they are dynamically selected by the specified control signals.

Finally, we can evaluate whether the communication architecture satisfies real-time constraints in an operation scenario defined by BFMs. In Fig. 2, after completing the second operation mode, BFM is triggered again by the third type of control signal "sync." The signal "sync" has a periodic nature and a BFM should complete the fixed amount of transfers within the period which is given by a real-time constraint. If BFM completes the transfers during the period, it becomes idle and waits for the next activation by "sync" signal. Otherwise, it transfers remaining data words at the next activation. This situation occurs due to the bus contention with other masters. The accurate modeling of the inter-dependency of masters is especially crucial to analyze bus contentions. The operation scenario in Fig. 2 is an example. We can create various scenarios by ordering the operation modes.

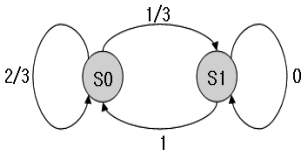
To summarize, we introduce three types of control signals. The first type of control signals initiates the operation of BFM and the second type is appropriate for sporadic events or interrupts from other components. The third type is intended for considering the real-time constraint given by a periodic signal. With these signals, we can model the inter-dependency of IPs and it is possible to create various operation scenarios for accurate performance analysis.

2.3 Stochastic Model for Computation Part

Trace-driven method is one option to implement the computation part. Its implementation is trivial and thus we omit the details. The other method is stochastic modeling. Even though the trace-driven method most accurately mimics the behavior of a master, its low flexibility becomes an issue when we explore the micro-architecture of master behavior. The key parameters to characterize the transfer behavior are transaction intervals and burst length. We build a state transition diagram for each operation mode and the state transition occurs in a probabilistic manner. First, we extract the states from the RTL simulation trace by quantizing transaction intervals. Transaction intervals are quantized into multiple groups and a state is assigned to each group. The resolution of quantization determines the number of states and affects the accuracy of the model. Figure 3 shows an example of state extraction from the simulation trace.

ID	time	burst length	State
0	0	8	S0
1	10	8	S0
2	20	4	S0
3	50	8	S1
4	60	8	S0
.....			

(a) A simulation trace.



(b) State transition diagram.

Fig. 3 A simulation trace and a state transition diagram.

Suppose that the quantization resolution is 10 time unit. In this case, the transaction intervals ranging from 0 to 10 belong to the same group and the transaction interval froms 11 to 20 belong to another group and so on. In Fig. 3, trace ID 0, 1, and 2 are in the same group because their transaction interval is 0 or 10 (the transaction interval of trace ID 0 is 0.). Trace ID 4 is also in the same group for same reason. Only trace ID 3 is in a different group because its transaction interval is 30. For each group we assign a state and the state transition probability can be computed as mentioned in Sect. 1 (transition counts from state A to state B over total transition counts from state A), after mapping state to each trace ID. In this example, the transition probability from S0 to S1 is 1/3 as shown in Fig. 3. For each state, we build a state transition diagram for burst length. Typically, burst length is a set of finite discrete numbers, thus we naturally create states with the appearing burst length numbers in each interval state. Similarly, we can extract the state diagrams for the read/write access patterns as well as address access patterns, even though we omit those in Fig. 3 for simplicity.

The transition probability is computed with the same method used in interval state transition diagram. The output control signals are also modeled in the same way if a BFM needs to model it. To summarize, our stochastic model consists of two levels of hierarchy. The upper level models interval variation and the lower level models burst length variation. Also, note that the transaction behavior is quite regular in a single operation mode due to the limited intervention of other control signals. In this case, we can achieve high flexibility while maintaining high accuracy comparable to trace-driven method.

2.4 Bus Functional Model Generation Flow

We show the BFM generation flow in Fig. 4. In the verification step of each IP component, we insert an AHB transaction monitor to collect the traces for BFM generation. The database from the monitor and a design configuration file are two inputs of our BFM generation tool. The database includes the information similar to a simulation trace shown in Fig. 3 for various operation modes. Supplementary information is given by a design configuration file. BFM generator instantiates a communication part which is common to every BFM and produces a stochastic model for each operation mode. Finally, it generates the control logic for control signals.

Figure 5 shows an example of a design configuration file. It specifies the information of control signals,

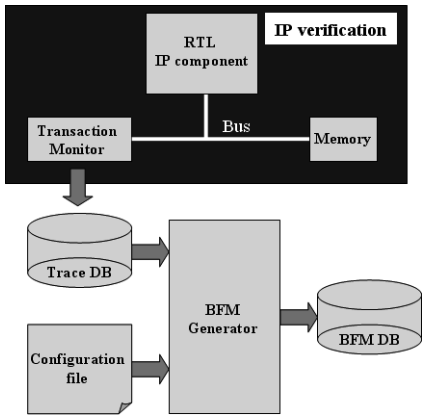


Fig. 4 BFM generation flow.

```
INIT = (init,1, 50)
ACTIVE = (active, 1, 30)
RTC = (sync, 1, 400, 200)
RW = READ
DATA_WIDTH = 32
ADDRESS = RANDOM
BURST_LENGTH = 8
```

Fig. 5 An example of a design configuration file.

read/write mode, data width, address pattern, and burst length of transactions. INIT and ACTIVE define "init" control signal and "active" control signal, respectively. They first define the triggering condition by the signal name and its logic state, and then specify the operation start time of connected IP components after the triggering condition is met (50 cycles in case of INIT). RTC specifies a signal which imposes a real-time constraint. It first defines the triggering conditions with the signal name and its logic state, and then specifies the value of real-time constraint in terms of clock cycles followed by the amount of data to be transferred. If RW, ADDRESS, and BURST_LENGTH are not specified, BFM generator produces a burst length state transition diagram mentioned in Sect. 2.3.

It is obvious that we can extract the signals which trigger the operation modes from the IP's external specification and their logic values. On the other hand, the triggering condition, the timing constraints (400 cycles), the amount of data to be transferred (200 cycles), and read/write access patterns are extracted from the system specification. For instance, the timing constraint and the amount of data to be transferred in the real-time constraint (RTC) description in Fig. 5 are the requirements from the system perspective, regardless of the corresponding IP's performance. If the IP cannot satisfy the system requirements, it will be reported during the simulation. The experimental results shown in Sect. 5 will address the performance analysis capability of our method more in detail.

3. Experimental Results

We conducted a set of experiments for a SoC designed for video processing systems to assess the impact of our

method. The target system consists of 16 master blocks including ARM9 which controls the overall system. The communication architecture is designed using AMBA2.0 AHB from ARM. The baseline communication architecture is single-layer AHB. In other words, all IP blocks and a memory controller are connected by a single AHB bus. The target system basically adopts a shared-memory communication scheme, thus the bus traffic is quite heavy and the communication architecture design is one of major concerns. The objectives of this experiment are as follows. First, we compare the accuracy of our method against RTL simulation and measure the speedup. Second, we need to explore several architectural choices to meet the given performance constraints with minimal effort. This is necessary to appreciate the benefits of our method. For the first objective, we compared the average and worst-case transaction latencies of IP blocks using our method to those from RTL simulation, where worst-case transaction latency means the longest latency among all observed latencies. We also implemented a performance monitor to measure the transaction latency of each IP block. The monitor is common to all IP blocks, since they have a common bus interface. Table 1 shows the experimental results for four timing-critical major blocks.

For other blocks not shown here, we observe the same level of accuracy and the overall average is 91.4%. It is worth to mention that the accuracy itself is over 90% and its deviation is not large, while the simulation speedup is more than 2500 in terms of CPS (Cycles Per Second). Such high accuracy was possible thanks to taking into account inter-dependency of IP blocks. Our method requires little modeling effort by reusing the RTL simulation trace and automated design flow. We could conclude that the baseline architecture is satisfying the given performance constraints from the latency perspective.

We also investigated the bus contention of IP blocks to exploit further performance improvement possibility. We vary the number of bus layers or improve the bus arbitration scheme. We measure the bus contention ratio which is defined as the waiting time of each IP block over a total time spent on data transfers. We measure the ratio using the performance monitor mentioned above. The simulation results show that the average bus contention ratio of all IP blocks is 16%, which is pretty low.

Table 1 Accuracy of the proposed method against RTL analysis.

IP blocks	Average latency accuracy	Worst-case latency accuracy
MPEG decoder	87.3%	91.9%
LCD controller	93.2%	93.7%
Post processor	89.5%	95.5%
Graphics accelerator	92.1%	100%
Average	90.5%	95.3%

Finally, we examined the data transfers of IP blocks constrained by real-time performance. In this simulation, we do not change the overall architecture, but we optimize the data buffer size of each IP to avoid underflow or overflow due to the violations of real-time constraints. We need a buffer for the remaining data when the data cannot be fully transferred for the given timing constraint and the buffered data should be transferred as shown in Fig. 2 (operation mode 2). However, it is not easy to determine the size of buffer due to the unpredictable inter-dependency of other IP blocks. From this experiment, we could found that the MPEG decoder violates the real-time constraint for 10% of total transfers and the maximum remaining data size is 5% of total data amount per transfer. This information is very important to avoid the real-time constraint violation or overdesign with a large buffer size.

4. Conclusion

We propose a scenario-aware bus functional modeling by considering the inter-play effect of IP blocks. Also, our method provides a stochastic method to model the computation part precisely. With our method, we could achieve over 90% accuracy for a SoC for video processing systems, while its speedup is over 2500 against RTL simulation. Ws also show we can apply it to determine several design choices — number of bus layers, arbitration scheme, and buffer size of each IP. Our method minimizes the modeling effort by reusing the RTL simulation trace and an automated design flow.

Acknowledgments

This work was partially supported by grant No. R01-2006-000-10156-0 from the Basic Research Program of the Korea Science & Engineering Foundation, by IT R&D Project funded by Korean Ministry of Information and Communications, and by IDEC (IC Design Education Center).

References

[1] A. Sangiovanni-Vincentelli, L. Carloni, F.D. Bernardinis, and M. Sgroi, “Benefits and challenges for platform-based design,” DAC, pp.409–414, 2004.
[2] S. Brini, D. Benjelloun, and F. Castanier, “A flexible virtual platform for computational and communication architecture exploration of DMT VDLS modems,” DATE, pp.164–169, 2003.
[3] J. Um, W.-C. Kwon, S. Hong, Y.-T. Kim, K.-M. Choi, J.-T. Kong, S.-K. Eo, and T. Kim, “A systematic IP and bus subsystem modeling for platform-based system design,” DATE, pp.560–564, 2006
[4] K. Lahiri, A. Raghunathan, and S. Dey, “System-level performance analysis for designing on-chip communication architecture,” IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol.20, no.6, pp.768–783, 2001.
[5] S. Kim, C. Im, and S. Ha, “Schedule-aware performance estimation of communication architecture for efficient design space exploration,” IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol.13, no.5, pp.539–552, May 2005.