УДК 004.415.538 DOI: 10.31114/2078-7707-2020-3-94-99

# Методика автоматизированного анализа производительности коммутационных сред с учетом структуры СнК и прикладных задач

К.А. Жезлов

АО «Научно-производственный центр «ЭЛВИС», kzhezlov@elvees.com

***Аннотация* — В статье описана инфраструктура и методика проверки подсистем коммутации на соответствие требованиям к производительности. Для реализации предлагаемой методики представлен генератор входных воздействий на основе графов и подходы к формированию базовых тестов производительности с его использованием. Также представлен набор метрик и критериев для анализа эффективности подсистем коммутации и приведены примеры их применения.**

***Ключевые слова* — система на кристалле, верификация, анализ производительности, сценарии использования, верификация на системном уровне, автономная верификация, графо-ориентированный подход, метрики производительности.**

## I. Введение

Современные системы на кристалле (СнК) состоят из большого числа разнородных сложно-функциональных блоков. В частности, к ним относятся процессоры разной архитектуры и периферийные блоки, объединенные системой коммутации. В круг задач верификации входит не только проверка корректности работы СнК на всех этапах проектирования, но и проверка её на удовлетворение требованиям к производительности, диктуемым целевыми задачами [1].

Основным способом проверки системы является автономная верификация СФ-блоков. Задача проверки функциональной корректности всей системы, как и анализ её эффективности при использовании модели всей СнК, являются очень ресурсозатратными как с точки зрения создания тестов, так и с точки зрения времени моделирования. Они требуют наличия рабочей модели всей системы, которая появляется слишком поздно для существенных переработок архитектуры в случае обнаружения проблем с производительностью системы. Обе эти задачи могут быть декомпозированы и выполнены на автономных окружениях при выполнении условия организации процесса передачи ограничений с верхнего уровня на уровень блоков и результатов характеризации отдельных блоков на уровень системы и других блоков [2].

Для обеспечения корректности работы системы по критерию эффективности маршрут разработки СнК должен быть ориентирован на раннюю локализацию проблем с производительностью и подтверждение её характеристик. Для решения этой задачи необходима унификация средств оценки производительности [3].

Производительность системы на кристалле во многом определяется её архитектурой и, в частности, архитектурой подсистемы коммутации, поэтому предпринимать шаги в отношении оценки всей системы на соответствие заданным требованиям необходимо как можно раньше.

Для оценки производительности системы, а также для поиска проблем с ней, критичны следующие факторы: тестовый сценарий, направленный на создание условий для измерения именно производительности, способ представления, формирования и исполнения этого сценария, подходящий для воспроизведения ошибок в дальнейшем, и методика оценки производительности, то есть какие параметры и каким образом рассчитываются [4, 5].

## II. Оценка производительности

Оценка производительности коммутационной среды, как правило, заключается в том, чтобы проверить, соответствует ли система заявленным требованиям по производительности.

Понятие производительности коммутационной среды обычно включает в себя следующие факторы:

1) Качество обслуживания (QoS – Quality of Service).
2) Полоса пропускания (bandwidth).
3) Пропуская способность (throughput).
4) Величина задержки (latency).
5) Количество одновременно выполняемых транзакций.
6) Арбитраж.

Основным этапом, с которого начинается оценка производительности, является характеризация нагрузки. Она включает в себя задачу описания шаблонов трафика и оценки его временных характеристик. То есть необходимо описать не только данные какого объема и откуда будут поступать в сеть, но и разреженность этих данных, объемы отдельных пакетов. В данном контексте речь идёт об абстрактном

трафике, не привязанном к протоколам передачи данных, поэтому на этапе характеризации не имеет значения тип используемых транзакций.

Основными метриками, по которым можно судить о производительности коммутационной среды в составе СнК, являются:

- ширина канала;
- пропускная способность;
- величина задержки.

Данный набор метрик является базовым и позволяет достаточно точно охарактеризовать производительность подсистемы коммутации, но не позволяет выявлять причины проблем производительности.

Коммутатор, будучи интегрированным в СнК, должен обеспечивать заданную эффективность выполнения класса целевых задач. Класс целевых задач определяется областью применения СнК (например, потоковая обработка видео/аудио).

Контролируемыми параметрами (с точки зрения задачи) являются:

1) время исполнения конкретной задачи;
2) время реакции системы на внешние события.

Факторы, потенциально влияющие на производительность системы и, следовательно, на контролируемые параметры, могут носить программный или аппаратный характер. Проблемы программного характера заключаются в неоптимизированности исполняемой задачи для данной архитектуры (в случае её удовлетворительной работы).

В случае аппаратной части можно выделить три основных проблемных места – ведущие и ведомые устройства, и сам коммутатор.

Коммутатор может стать причиной низкой производительности системы в следующих случаях:

- когда он не обеспечивает необходимой ведущим и ведомым устройствам пропускной способности;
- неправильно настроен арбитраж.

В качестве ведущих устройств, как правило, выступают CPU. Ведущие устройства потенциально могут понижать производительность в случае неоптимизированности их настроек для работы с памятью.

В качестве ведомых устройств чаще всего выступают различные виды памяти. В данном случае проблемой может являться низкая скорость её работы и неоптимизированность внутренней коммутации.

## III. Графо-ориентированный подход к формированию тестовых воздействий

Наиболее удобным для решения задач формирования и исполнения тестового сценария является его представление в виде графа. Такой подход позволяет формировать сценарий, оперирующий транзакциями или даже потоками данных, что позволяет перенести описание теста на более высокий уровень абстракции. Независимость от внутренней архитектуры блоков повышает степень повторного использования кода тестовых сценариев и облегчает их модификацию. При этом детерминированность тестового сценария позволяет однозначно воспроизвести ситуацию, в которой была обнаружена ошибка работы системы, и определить конкретное событие, ставшее её причиной.

В описываемом подходе тестовый сценарий представляет собой дерево транзакций: вершинами графа являются сами транзакции, а ребрами – зависимости между ними.

В состав предлагаемого инструмента входят два основных блока – генератор графов (Python) и их проигрыватель (System Verilog). На рис. 1 представлена его структурная схема.

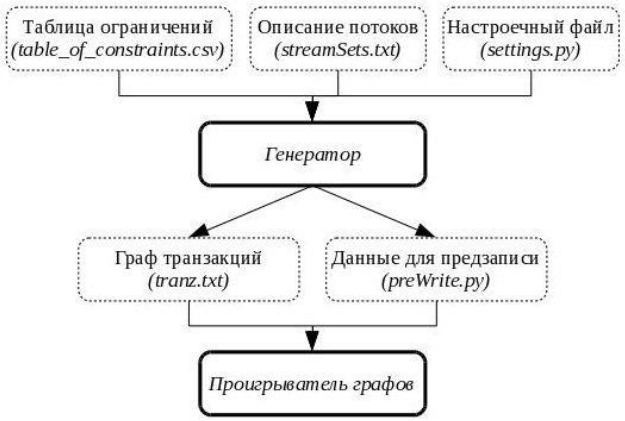


**Рис. 1. Структурная схема генератора и проигрывателя графовых тестовых сценариев**

Инструмент позволяет генерировать тесты следующих типов: тесты для проверки карты памяти, синтетические тесты для оценки производительности, тесты на основе сценариев использования устройства. Входными данными для блока генератора являются: таблица ограничений устройства, которая включает в себя карту памяти, матрицу коммутации и ограничения для master и slave устройств; настроечный пользовательский файл; текстовый файл с описанием потоков данных, на основе которого будет построен граф транзакций. Выходными данными являются файл с текстовым описанием графа транзакций и файл с данными и адресами для предварительной записи в память.

Описание потоков данных представляет собой определение следующих величин: номера master и slave устройств, осуществляющих приём и передачу данных, операция (чтение или запись), количество одновременно отправляемых транзакций, длина и ширина транзакций, объем отправляемых данных и скорость их отправки, диапазон адресов и имена потоков, которые будут запущены после этого. Эти параметры позволяют конструировать сложные

тестовые сценарии, основанные на зависящих друг от друга передачах массивов данных.

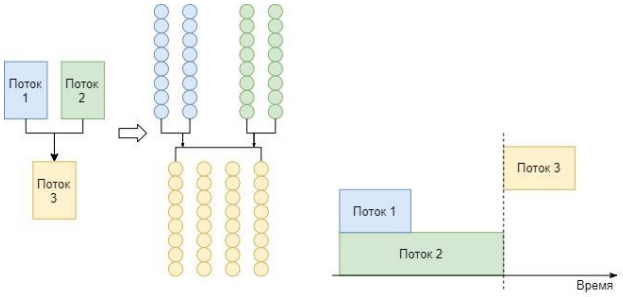


**Рис. 2. Формирование и проигрывание графа транзакций**

На рис. 2 представлен пример формирования графа транзакций по описанию потоков. В нем рассматривается создание трех потоков, при этом третий зависит от первых двух и должен запуститься только тогда, когда выполнятся все транзакции из Strm1 и Strm2. В первых двух потоках транзакции отправляются по 2 одновременно, в третьем – по 4, и передаётся вдвое больший объем данных. Как видно из примера, граф определяет лишь последовательность транзакций и скорость инжекции, то есть при моделировании два одинаковых одновременных потока могут быть исполнены за разное время.

Таким образом, подобные тестовые сценарии, описанные на высоком уровне абстракции, позволяют с достаточной точностью имитировать трафик, характерный для различных устройств и сценариев их использования, с учетом реальной динамики передачи управления тем или иным фазам сценария. Высокоуровневое описание позволяет использовать графовый подход не только для тестов, направленных на оценку производительности, но и для функционального контроля архитектуры системы или подсистемы на ранних этапах разработки (в частности на автономном окружении коммутационной среды).

## IV. Метрики и критерии оценки производительности

Для вычисления предлагаемых в данной работе метрик производительности используются следующие характеристики транзакций и условные обозначения:

- $T_n$ – текущая транзакция;
- $Tsize_n$ – объем текущей транзакции;
- $start_n$ – время начала текущей транзакции;
- $end_n$ – время завершения текущей транзакции;
- $end_{n-1}$ – время завершения предыдущей транзакции;
- $t$ – длина временного промежутка, на котором производится измерение.

Для оценки производительности коммутаторов предлагается использовать следующие метрики:

1) **Время исполнения транзакции** (такт, с) – латентность, время между началом передачи транзакции и полным ее приёмом.

$$L_n = end_n - start_n$$

2) **Средняя скорость передачи данных** (Байт/такт, Байт/с) – показывает фактическую загрузку канала.

$$Vd_{avg} = \frac{\sum Tsize_n}{t}$$

3) **Локальная пропускная способность** (Байт/такт, Байт/с) – максимальный трафик, принимаемый каналом.

$$C_n = \frac{Tsize_n}{end_n - end_{n-1}}$$

4) **Скорость исполнения транзакции** (Байт/такт, Байт/с) – объем данных, переданный в течение времени исполнения одной транзакции.

$$E_n = \frac{Tsize_n}{end_n - start_n}$$

5) **Незавершённые транзакции** (шт) – количество транзакций, выполняющихся параллельно.

$$O_n = \sum T_m + 1,$$

для $T_m$ таких, что $start_n \le start_m \le end_n$.

Расчет значений метрик производится на основе логов симуляции, из которых экстрагируются такие значения, как длина транзакции, время её начала и конца. Может производиться как расчет значений на различных временных промежутках (при необходимости анализа зависимости значения метрики от времени), так и на всём временном отрезке моделирования для получения интегральных значений. В случае вычисления значений метрик на нескольких временных отрезках нужно учитывать, что увеличение количества отрезков, на которые разбивается всё время симуляции, увеличивает точность расчёта, но увеличивает затрачиваемое время.

Базовый анализ производительности производится на основе пропускной способности и времени исполнения транзакции. Исходя из соотношения этих величин и характера трафика, можно сделать выводы об эффективности системы.

Перед определением критериев необходимо ввести ряд условных обозначений:

- $Vd_{ref}$ – эталонное значение пропускной способности.
- $K_c$ – коэффициент допустимого отклонения локальной пропускной способности, $0 \le K_e \le 1$.
- $K_e$ – коэффициент допустимого отклонения скорости исполнения транзакций, $0 \le K_e \le 1$.

Ниже перечислены основные критерии, по которым производится оценка.

1) $Vd_{avg} < Vd_{ref}$ – устройство не обеспечивает необходимой скорости передачи данных.

2) $C_{avg} < K_c C_{\max}$ – локальная пропускная способность значительно ниже максимальной. Потенциально устройство может работать быстрее.

3) $E_{avg} < K_e E_{\max}$ – транзакции исполняются медленнее, чем потенциально могли бы. Потенциально устройство может работать быстрее.

4) $C_{\max} < Vd_{ref}$ – максимальная локальная пропускная способность ниже ожидаемой пропускной способности. Нужная скорость передачи данных не достигается.

5) $C_{avg} \approx E_{\max}$ – транзакции передаются плотным потоком. Канал передачи полностью загружен. В противном случае устройство не создаёт необходимой нагрузки.

6) $E_{avg}(MS) < E_{avg}(SL)$ – ведущее устройство медленнее обрабатывает транзакции.

7) $E_{avg}(SL) < E_{avg}(MS)$ – ведомое устройство медленнее обрабатывает транзакции.

Важно помнить, что скорость передачи данных и время исполнения транзакции не имеют прямой или обратной зависимости. Уменьшение скорости передачи данных и уменьшение времени исполнения транзакции говорит о том, что транзакции исполняются за меньшее время, но промежутки между ними увеличились.

## V. Базовые тесты производительности

Базовыми тестами производительности, являются тест латентности, пропускной способности и количества одновременно исполняемых транзакций (outstanding). Эти тесты имеет смысл проводить, как только RTL-описание коммутационной инфраструктуры прошло базовую функциональную проверку на соответствие карте памяти и карте регистров. Для каждого из базовых тестов производительности необходим его запуск для каждой пары master-slave в соответствие с таблицей коммутации. Кроме того, необходима отдельная проверка для операций чтения и записи.

Отслеживаемой метрикой для теста латентности является среднее время исполнения транзакции, так как она показывает время между началом передачи транзакции и полным ее приёмом. Добавление дополнительных задержек к сигналам для данного теста не требуется. Для получения корректных значений необходима передача транзакций с минимальной длиной, но количество транзакций должно хотя бы в два раза превышать количество временных отрезков, на которых будет проводиться измерение. Эмпирически было выяснено, что в среднем необходима передача около двухсот транзакций. Кроме того, транзакции должны отправляться по одной.

Тест пропускной способности проводится также отдельно для каждой пары master-slave и операций чтения и записи. Отслеживаемой метрикой для данного теста является средняя локальная пропускная способность. Для этого теста необходимо максимально загрузить используемый канал подсистемы коммутации, поэтому транзакции максимальной длины должны передаваться в сеть с максимальной скоростью (то есть с нулевой паузой между транзакциями) и с максимальным количеством одновременно исполняемых транзакций. Данный тест необходимо проводить как с нулевыми задержками для оценки максимальных возможностей коммутационной инфраструктуры, так и с реальными задержками для всех устройств для получения представления о том, насколько близка работа системы к максимальным значениям при идеальных условиях.

Тест одновременно исполняемых транзакций производится при минимальных размерах длины и ширины транзакций, а сами транзакции должны отправляться с максимальной скоростью. Для такой проверки необходимо одновременно отправить в сеть количество транзакций большее, чем может передать master-устройство и принять slave-устройство. Для получения корректных результатов необходима установка дополнительных задержек на ответные сигналы (например, RVALID) для того, чтобы искусственно увеличить время исполнения транзакций, то есть задержать момент «valid handshake» и начало фазы передачи данных. Величина задержки должна быть такой, чтобы до завершения первой транзакции master-устройство могло передать адреса для всех отправляемых транзакций. То есть если предположить, что фаза передачи одного адреса занимает один такт, а в рамках теста необходимо отправить 300 транзакций, то величина задержек должна составлять минимум 300 тактов. Кроме того, для данного теста необходимо управление полем id транзакций со стороны теста, так как от количества id, зарезервированных для данного порта, зависит и количество outstanding-транзакций. То есть при составлении теста необходимо учитывать ожидаемое значение того, сколько транзакций с одинаковыми id данный порт может отправить, не дожидаясь их завершения.

Данный тест может проводиться в двух формах – оценка максимального количества outstanding-транзакций и оценка их количества в зависимости от времени. В первом случае необходимо вычислить количество транзакций, которые начались до того, как успела закончиться первая запущенная транзакция. Это позволяет не только оценить максимальное количество одновременно запущенных транзакций, но выявить ошибки с неверной настроенной глубиной входных или выходных буферов устройств. Для второго случая необходима проверка методом скользящего окна – нужно для каждого отрезка времени вычислить количество транзакций, которые были запущены в этом отрезке или же были запущены раньше, но не завершились к моменту начала этого отрезка.

## VI. Пример оценки производительности системы

При сравнении средней локальной пропускной способности и средней скорости исполнения транзакции можно выявить причину ограничения пропускной способности. Если пропускная способность ограничивается со стороны коммутационной логики, то есть создаваемая мастером нагрузка выше фактических возможностей коммутатора в данной ситуации, то средняя локальная пропускная способность будет очень близка к максимальной скорости исполнения транзакций. При этом получаемое значение пропускной способности будет ниже, чем рассчитанное при проектировании. Эта ситуация говорит о том, что все транзакции исполняются максимально быстро и идут с минимальными паузами. На рис. 3 представлен график, иллюстрирующий подобную ситуацию.

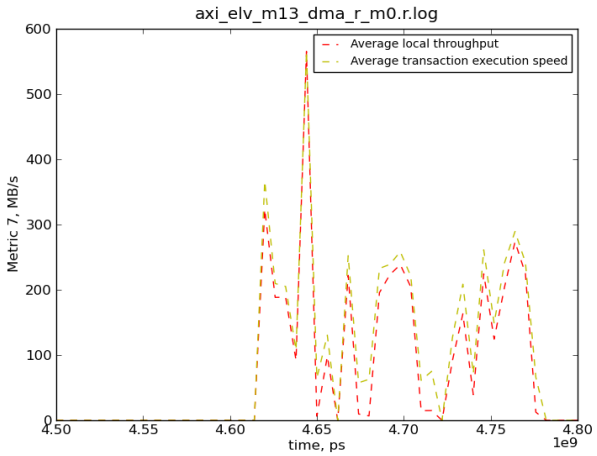


**Рис. 3. Ограничение пропускной способности со стороны коммутационной логики**

Пропускная способность может быть ограничена со стороны мастера, то есть создаваемая им нагрузка ниже фактических возможностей коммутатора в текущей ситуации. Причиной этого могут быть некорректно сформированные тестовые воздействия, например, передача слишком коротких слов, неиспользование всех outstanding-транзакций или неверная настройка генерации id транзакций. Если тест сформирован корректно, то причинами несоответствия пропускной способности требованиям может быть неправильно настроенный арбитраж или некорректная работа master-устройства. В данном случае средняя скорость исполнения транзакций будет значительно превышать среднюю локальную пропускную способность, что говорит о том, что транзакции исполняются быстро, но идут с большими паузами, увеличивая время простоя системы. На рис. 4 представлен пример ограничения пропускной способности со стороны мастера.

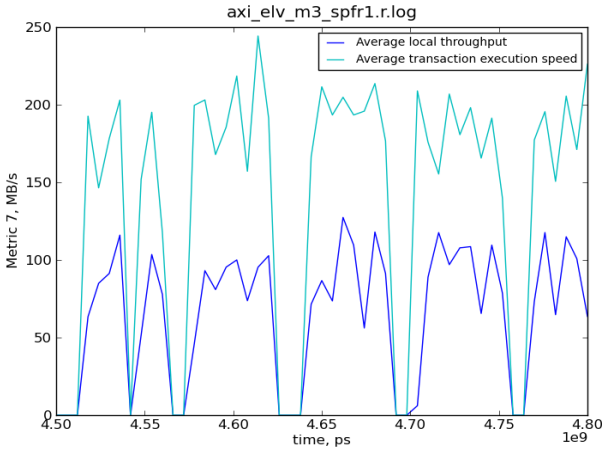


**Рис. 4. Ограничение пропускной способности со стороны мастера**

## Литература


[1] Mehta A.B. ASIC/SoC Functional Design Verification / A.B. Mehta // Springer International Publishing. 2018.

[2] Головина Е.К. и др. Метод создания и отладки комплексных тестов для функциональной верификации СнК, ориентированный на их повторное использование на всех этапах проектировании // Проблемы разработки перспективных микро-и наноэлектронных систем (МЭС). 2014. №. 2. С. 45-50.

[3] Automation of Test Environment Creation Aimed at IP-cores and SoC Development, Verification and Performance Analysis / Zhezlov K.A., Kolbasov Y.S., Kozlov A.O., Nikolaev A.V., Putrya F.M., Frolova S.E. // Проблемы разработки перспективных микро- и наноэлектронных систем (МЭС). 2017. № 2. С. 16-21.

[4] Performance Evaluation of NoC-based Multicore Systems: From Traffic Analysis to NoC Latency Modelling / Z. Qian, P. Bogdan, CY. Tsui, R. Marculescu // ACM Transactions on Design Automation of Electronic Systems (TODAES). 2016. Т. 21. №. 3. С. 52.

[5] Effects of the NoC Architecture in the Performance of NoC-based MPSoCs / D.R. Silva, B.S. Oliveria, F.G. Moraes // 2014 21st IEEE International Conference on Electronics, Circuits and Systems (ICECS). IEEE, 2014. С. 431-434.

# Methodology of Automated Performance Analysis of SoC Interconnect Subsystems, According to SoC Structure and Application

K.A. Zhezlov

Joint Stock Company Research and Development Center «ELVEES», kzhezlov@elvees.com

***Abstract* — The paper describes the infrastructure and methods for checking an interconnect subsystem as a part of a SoC for compliance with performance requirements. The graph-based test scenarios generator is suggested as a part of the infrastructure being mentioned. Also the approach for developing basic performance tests is described. There is a description of a set of performance metrics and criteria, which are tested on several examples, detecting performance problems. The following factors are critical for evaluating the performance of a system, as well as for searching for its bottlenecks: a test scenario aimed at creating conditions for measuring performance; a way of representing, creating and executing this scenario, suitable for reproducing errors in the future; performance evaluation methodic. The most convenient for solving the problems of the creation and execution of the test scenario is its presentation as a graph. This approach allows you creation of scenario that operates with transactions or even data flows, which allows you transferring the test description to a higher level of abstraction. The graph-based approach describes a test scenario as an acyclic directed graph of transactions, the vertices of the graph are the transactions themselves, and the edges are the dependencies between them. This approach makes the test scenario reusable and error reproducible. Calculation of performance metrics and evaluation of criteria make it easier to evaluate the interconnect subsystem performance and find its bottlenecks.**

***Keywords* — SoC, verification, performance analysis, usage scenarios, system level verification, standalone verification, high-level model, graph-based approach, performance metrics**

## REFERENCES

[1] Mehta A.B. ASIC/SoC Functional Design Verification / A.B. Mehta // Springer International Publishing. 2018.

[2] Golovina E.K. Metod sozdaniya i otladki kompleksnykh testov dlya funktsional'noy verifikatsii SnK, oriyentirovannyy na ikh povtornoye ispol'zovaniye na vsekh etapakh proyektirovanii (Method for creating and debugging complex tests for functional verification of SoCs, focused on their reuse at all stages of design) // Problemy razrabotki perspektivnykh mikro-i nanoelektronnykh sistem (MES). 2014. №. 2. S. 45-50 (in Russian).

[3] Automation of Test Environment Creation Aimed at IP-cores and SoC Development, Verification and Performance Analysis. Zhezlov K.A., Kolbasov Y.S., Kozlov A.O., Nikolaev A.V., Putrya F.M., Frolova S.E. // Проблемы разработки перспективных микро- и наноэлектронных систем (МЭС). 2017. № 2. С. 16-21 (in Russian).

[4] Performance Evaluation of NoC-based Multicore Systems: From Traffic Analysis to NoC Latency Modelling / Z. Qian, P. Bogdan, CY. Tsui, R. Marculescu // ACM Transactions on Design Automation of Electronic Systems (TODAES). 2016. Т. 21. №. 3. С. 52.

[5] Effects of the NoC Architecture in the Performance of NoC-based MPSoCs / D.R. Silva, B.S. Oliveria, F.G. Moraes // 2014 21st IEEE International Conference on Electronics, Circuits and Systems (ICECS). IEEE, 2014. С. 431-434.